



工業技術研究院

Industrial Technology
Research Institute

虛實整合製造系統軟體開發平台 使用手冊

VMX Host Manual

版本: 2.5.0

王培寧

Wang, Pei-Ning

pnwang@itri.org.tw

工業技術研究院
智慧機械科技中心



目錄

第一章、介紹.....	4
第二章、安裝與環境.....	6
2.1 安裝 VMX.....	6
2.2 環境.....	9
2.2.1 CNC 連線設定	11
2.2.2 DAQ 設定.....	14
2.2.3 服務設定.....	15
2.2.4 VMX 系統設定.....	15
第三章、軟體開發.....	17
3.1 APP 掛載機制	17
步驟 1、新增 Windows Form 專案	17
步驟 2、加入相關參考組件	17
步驟 3、實作 APP 的介面.....	18
步驟 4、匯出成 DLL 檔案	19
3.2 APP 的除錯環境設定	21
3.3 CNC 轉接器的應用	23
3.3.1 基本操作.....	23
3.3.2 程式碼範例.....	24
3.4 DAQ 轉接器的應用	25
3.4.1 設定 DAQ 轉接器.....	25
3.4.2 使用 DAQ 轉接器.....	25
3.4.3 DAQ 設備的運作概要.....	26
3.4.4 使用 DAQ 緩衝.....	27
3.5 歷程記錄器的應用	29
3.6 APP 間的溝通機制	30
第四章、可自製的通訊/操作介面	32
4.1 CNC 通訊介面	32
4.1.1 介面繼承.....	33
4.1.2 介面定義與實作方法.....	34
4.1.3 測試.....	37



第五章、應用案例.....	39
5.1 CNC 資料擷取範例	39
5.2 訊號擷取範例	43

第一章、介紹

VMX 是財團法人工業技術研究院（以下稱「工研院」）針對智慧機械系統整合所研發製作的軟體平台。VMX 的開發精神可由其三個英文字母代表，分別是

V : Virtual or Cyber

M : Multiple

X : Change or Extend

VMX 的開發目的即是為了創造具拓展功能的多功能虛擬平台。

VMX 早期是工研院開發防碰撞系統(Collision Avoidance System, CAS)時的底層產物。當時的目的是建立一套統合不同 CNC 控制器軟體通訊介面的 API，讓防碰撞系統不需要針對不同的工具機 CNC 準備特定的通訊介面，加速軟體技術應用於各式工具機 CNC 上；結果成功地讓防碰撞系統在發那科(Fanuc)、海德漢(Heidenhain)與西門子(Siemens)等 CNC 上運行。緊接著，顫振迴避技術(Chatter Avoidance Technology)也透過該 API 在前述的 CNC 上使用。同時期，台灣內部的法人單位陸續向業界推出類似的技術與相關產品，例如資策會的 ServoBox 與精機中心的 SkyMars，但工研院的 VMX 相關技術仍未對外推出，主要是由於開發成員注意到客戶與開發者的潛在需求。

在防碰撞系統與顫振迴避技術的應用時，除了技術本身之外，客戶最常詢問的問題有二類。第一類是如何將感測器的資源共享給其他單位的技術，例如振動監控、預兆診斷、虛擬檢測等需要感測訊號的軟體技術。這是由於感測器的存取資源通常會被特定的軟體獨佔，造成客戶設備上需要安裝多個類似的感測器，並且造成安裝成本的提升。

另一類是異常訊息如何即時發送到機台與管理系統，例如碰撞發生，生產管理者是否能儘早得知。VMX 開發團隊注意到，最佳化整合資源以及連結生產管理系統是客戶的主要需求，任何智慧機械相關的軟體或技術都應該考慮到資源整合與管理需求，因此決定研究與開發新架構的機電軟體整合平台，並正式以 VMX 命名這一套產品。從此開始，VMX 從一套工研院內部開發的 API 蛻變為一套連結未來需求與多種用途的軟體整合平台。

VMX 軟體整合平台的主要特色如下所述：

1. 類似 OPC 架構的可拓展 CNC 通訊介面。

使用者可透過資料路徑存取資料，因此使用者僅需要查詢路徑的字串(String)與資料型別(Type)，加速軟體開發速度。另一方面，此通訊介面採用半開放式的開發模式，開發者可以開發自己需求的 CNC 通訊介面，並且將該介面掛載至 VMX 平台上。目前支援的 CNC 有發那科、海德漢、西門子、三菱、GMC、FAGOR 等。

2. 類似 OPC 架構的可拓展設備通訊介面。

除了 CNC 之外，工廠內有許多其他自動化設備，例如 PLC、HMI、機器人、運動控制卡等。VMX 提供半開放式的開發模式，開發者可以開發自己需求的相關通訊介面，供後續開發的應用程式(Application, APP)使用。

3. 感測資源共享機制。

當設備安裝一個感測器時，它的即時訊號可以讓 VMX 平台上任意的應用程式(Application, APP)使用。舉例而言，顫振偵測與機台故障檢測可以共享同一個加速規的訊號。

4. 預先準備好的 SQL 資料庫。

目前業者常用的生產管理系統多半是基於 SQL 的系統(SQL-based System)。直接提供簡易的 SQL 資料庫，可以縮短資料收集單位（例如 VMX）與生產管理系統相互連結的整合時間，並且提供資料備援機制，避免因網路或設備故障而造成資料短缺，確保資料收集的可靠度及完整性。

5. APP 市集

由於台灣設備製造業者多為中小企業，具有十分優異的硬體開發能量，但是當設備搭載軟體出售到世界各地時，軟體更新就變成十分困難的課題。VMX 系統包含 APP 市集(App Store)，無論是 VMX 本身的升級或 APP 的更新，均可以透過雲端伺服器處理，避免軟體更新的困擾。

6. 雲端服務

VMX 除了可以掛載 APP 之外，也能安裝服務模組(Service Module)，例如自動資料收集(Auto Data Collection, AutoDC)服務。這些背景服務可以提供資料收集、監控、紀錄與回報等共通性資訊處理機制。一方面可以減少 APP 開發時的負擔，另一方面記錄與回傳機台異常時的資料，讓業者能夠進行雲端服務。

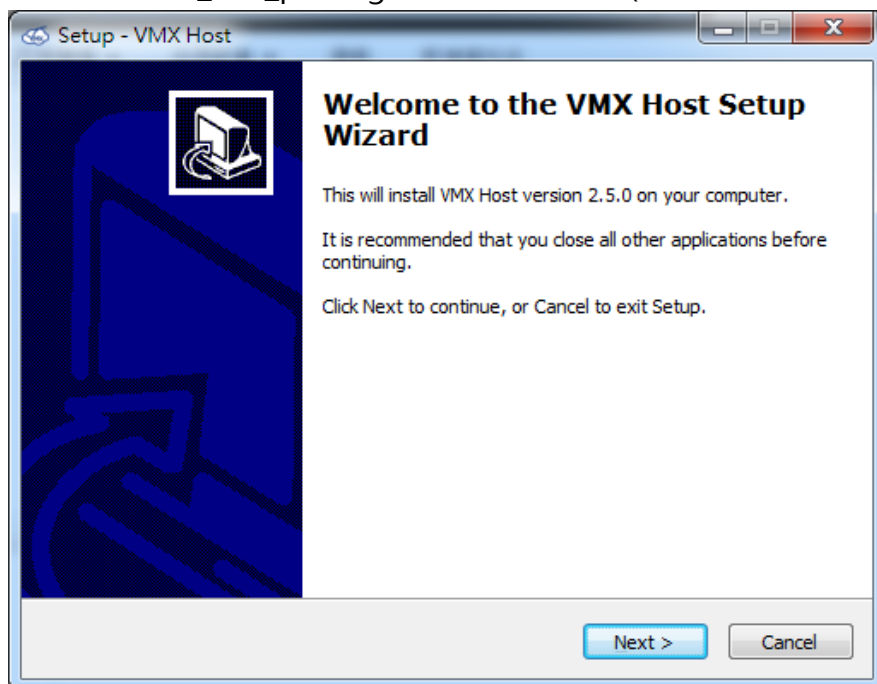
7. 其他模組

VMX 不僅由工研院智慧機械科技中心開發，也包含其他工研院單位的應用模組，例如 OPC UA 伺服器服務、與 PaaS 連接、可安裝於 NAS 系統上、統計製程管理 (SPC) 模組等相關技術。

第二章、安裝與環境

2.1 安裝 VMX

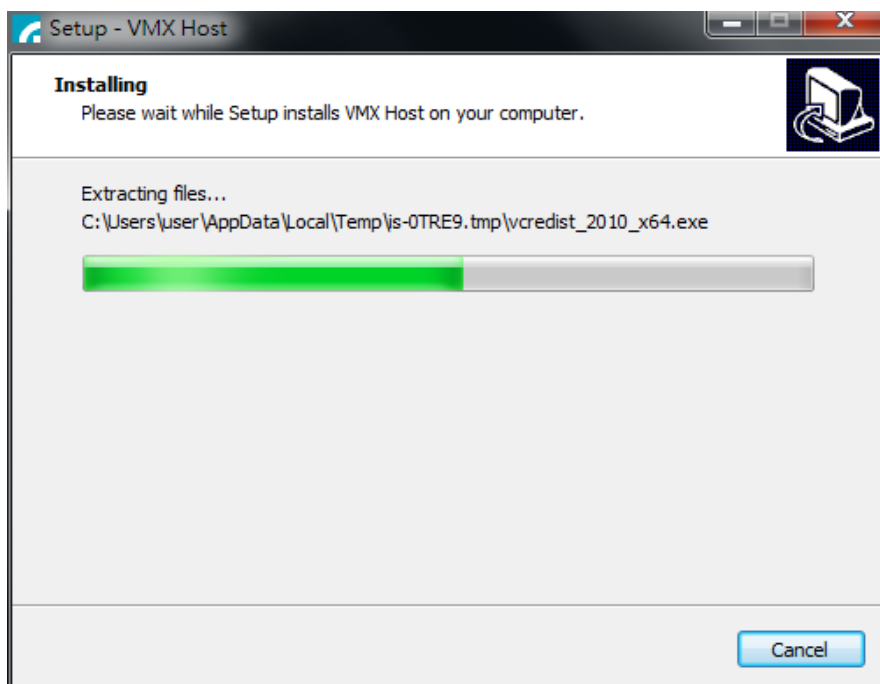
1. 執行 vmx250_x64_package.exe 安裝 VMX (實際檔名因版本而異)。



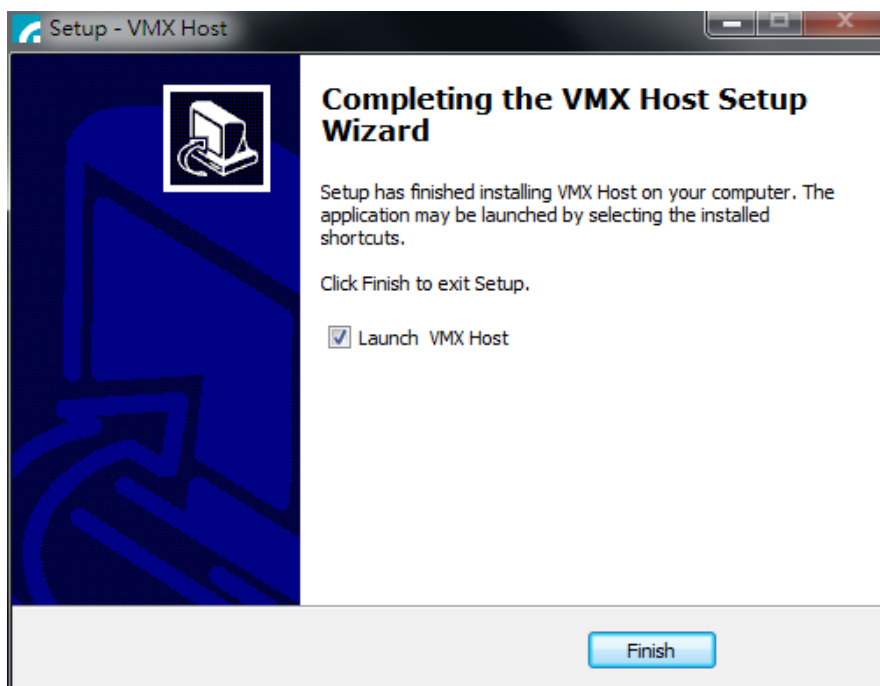
2. 開始安裝，必須同意 EULA 才能繼續安裝。



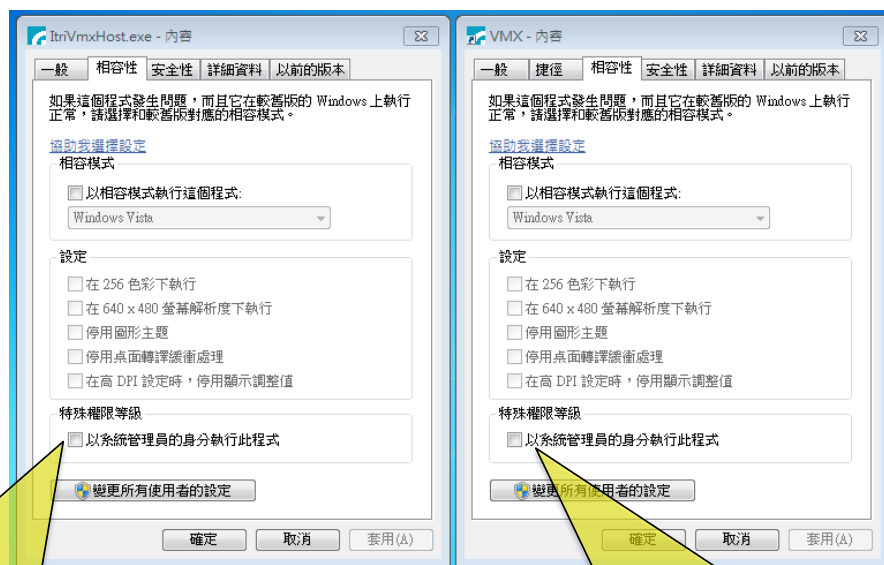
3. 如果沒有安裝.Net Frameworks 或相關套件，可能會自動安裝或更新。



4. 安裝完成



5. 如果 VMX 安裝在特殊目錄，例如預設的 C:\Program files，會需要系統管理員權限，有二種開啟辦法。若沒有啟用系統管理員權限，可能會導致非預期的程式關閉。如果要把 Visual Studio 輸出的結果在特殊目錄中執行，也會需要以系統管理員權限啟動 Visual Studio。

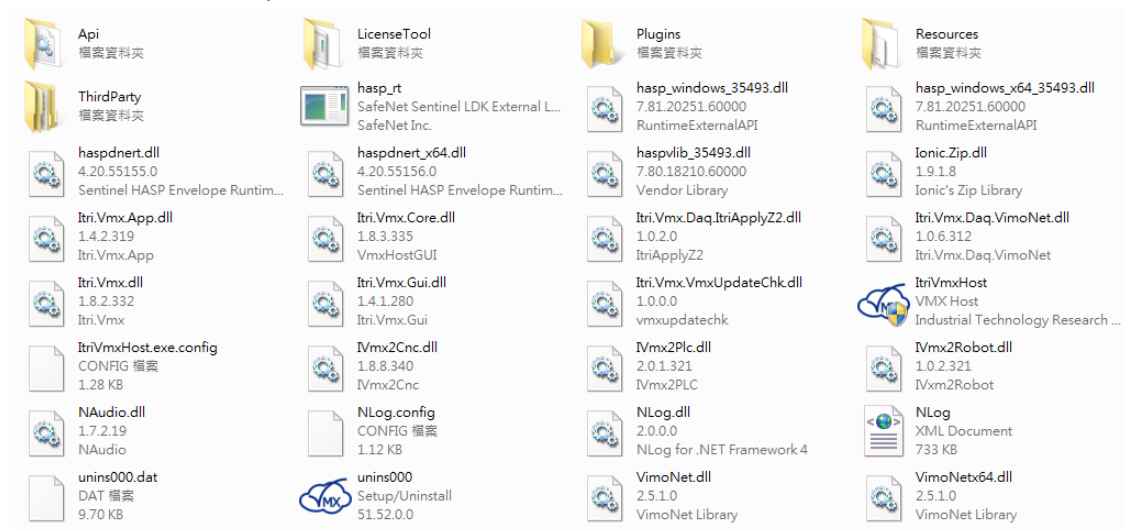


在安裝目錄下的
ltriVmxHost.exe提供系統管理
員權限

在捷徑的內容內提供系統管理
員權限

2.2 環境

開啟 VMX 資料夾 (預設路徑為 C:\Program Files\ITRI\VMX Host)，可以看到以下的資料夾結構 (實際內容可能會有不同)：



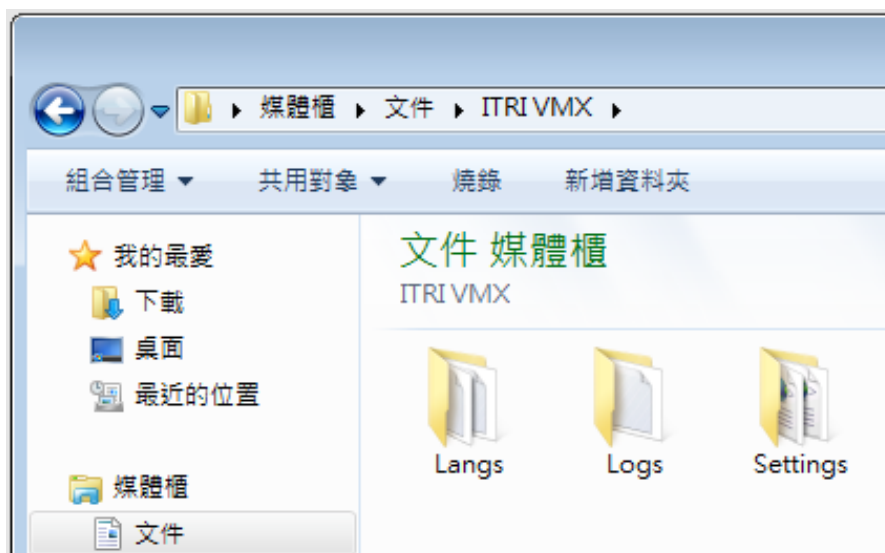
執行 ItriVmxHost.exe，即可進入 VMX 程式畫面：



右側工具列包含下列功能：

	設定(Configuration)：可在此編輯 CNC、DAQ、VMX 系統等相關設定
	記錄(Logs)：開啟 VMX 記錄檔資料夾
	安裝(Install)：由 ZIP 檔安裝 APP
	市集(App Store)：開啟 APP 市集，可線上下載 APP (尚未開放)
 	預設 APP(Default APP)：可直接進入預設 APP 畫面。 - 若無設定預設 APP，為左側圖示，此時點選並無作用 - 若已設定預設 APP，則會顯示 APP 圖示，如右側圖示 - 於主畫面的 APP 上點右鍵選擇 Set as default app 設定預設 APP
	程式集(Programs)：返回 VMX 主畫面

VMX 的設定和記錄檔的預設存放於 VMX 工作目錄，預設路徑為 Documents/ITRI VMX：



其中 Settings 內有存放 CNC 和 DAQ 等設定檔，Logs 內有錯誤訊息的記錄檔，Langs 內有語言設定檔。

VMX 支援的控制器通訊介面與 API 資料夾內的組件有關，在『第四章、客制化通訊介面』將會詳述此一機制。可在 Settings 資料夾內新增、編輯或刪除 XML 檔案，更改 VMX 對各種設備的連接環境。VMX 可以支援多機台連線。由於通訊介面對應之 XML 格式是由組件開發者決定，使用者僅能修改 XML 節點的資料。為了確保 XML 格式與通訊介面的要求一致，建議依照 2.2.1 的說明產生預設格式的 XML 檔案，之後再根據需求修改 XML 的節點資料。

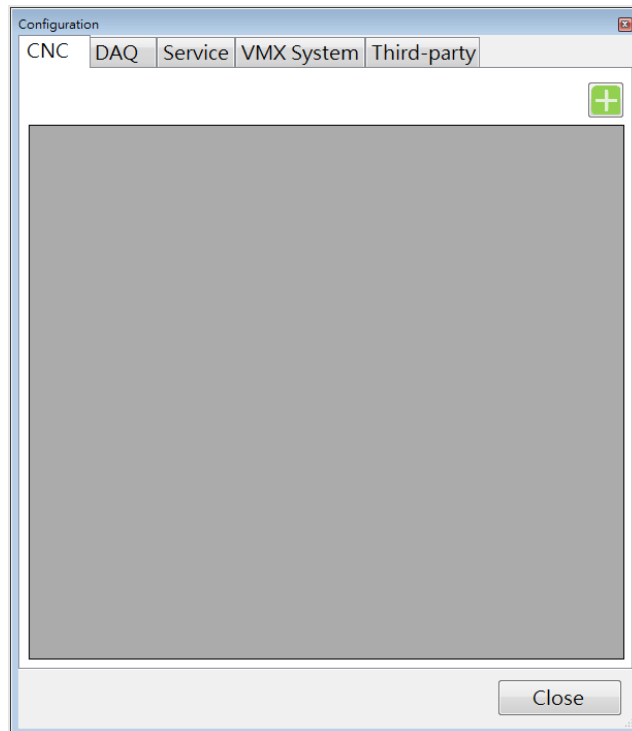
2.2.1 CNC 連線設定


以下示範連線 Fanuc 30i 系列的 CNC 裝置的步驟：

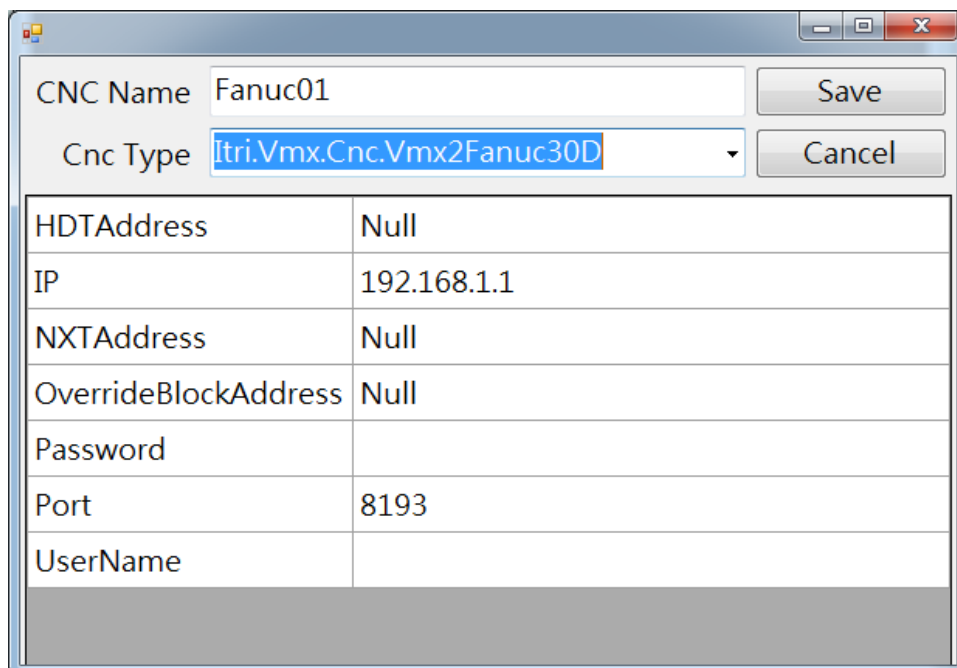
1. 透過 VMX 安裝功能  載入 VMX 對 CNC 通訊介面檔案包 (ZIP)



2. 點選右上角的  進入設定畫面

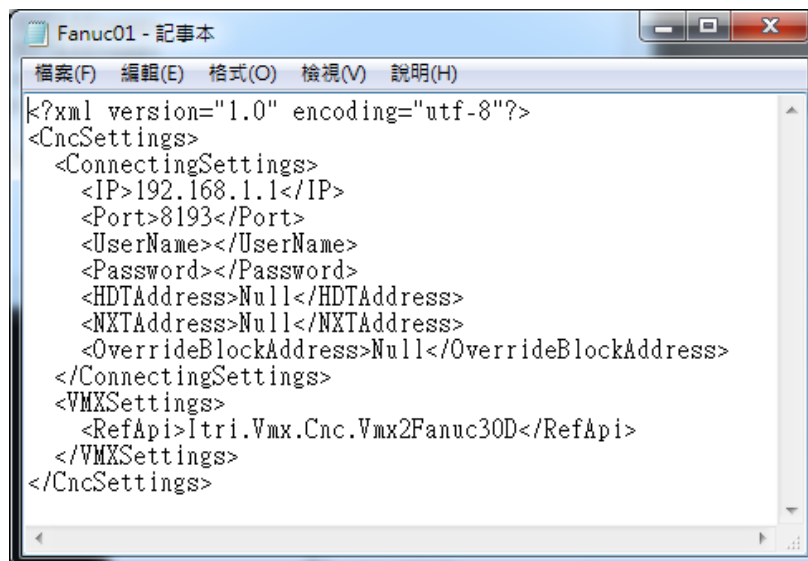



3. 點選 CNC 標籤進入 CNC 設定畫面，並點選右上角的綠色加號  開啟新增設定畫面，接著輸入 CNC 名稱(可自由取名)並選擇 CNC 類型(可選擇先前置入 API 中的通訊介面)。此時下方表格會自動顯示連線相關資訊，可於此變更設定值。設定完成後點選存檔(Save)，即可將 CNC 設定保存成 XML 檔案。

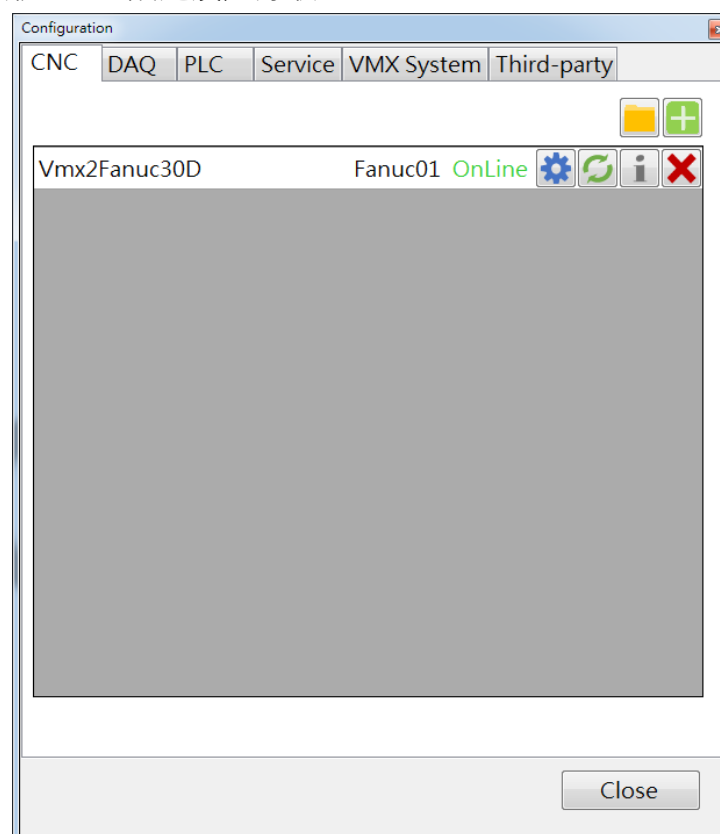






CNC Name	Fanuc01	Save
Cnc Type	Itri.Vmx.Cnc.Vmx2Fanuc30D	Cancel
HDTAddress	Null	
IP	192.168.1.1	
NXTAddress	Null	
OverrideBlockAddress	Null	
Password		
Port	8193	
UserName		

4. 可於 Settings\cnc 中檢視 CNC 的設定檔，使用者亦可於此更改 CNC 連線設定內容。



5. 重新啟動 VMX 並再次點選  進入設定畫面，點選 CNC 標籤進入 CNC 設定頁面，即可於中央的 CNC 列表中檢視 CNC 設定及連線狀態。



6. 點選  圖示可檢視和編輯 CNC 設定內容、點選  可重新整理 CNC 連線狀態、點選  可快速確認 CNC 之狀態、座標、當程式名稱等簡易資料、點選  可刪除 CNC(會一併刪除設定檔)。

2.2.2 DAQ 設定

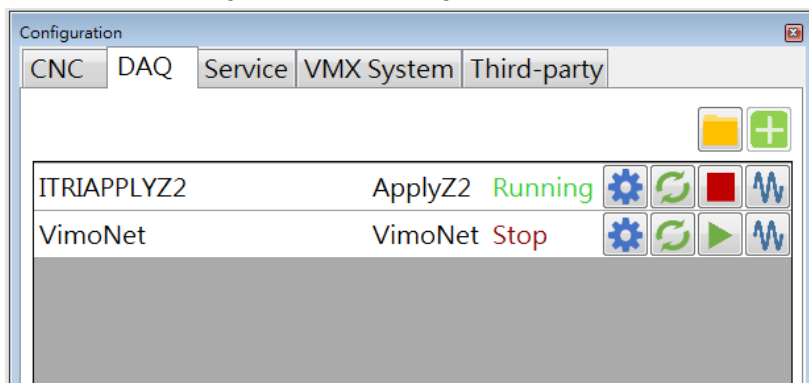
於 Settings 資料夾中開啟與編輯 DAQ.XML 檔案。VMX 支援多台資料擷取設備的使用。可透過 VMX API 的 DaqAdaptors 陣列取得連線中設備的參數，其中陣列元素 0 是第 1 台設備，陣列元素 1 是第 2 台設備，依此類推。







```

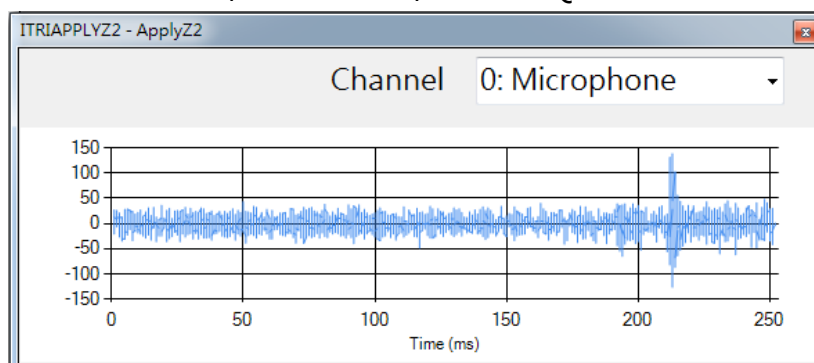
1 <?xml version="1.0" encoding="utf-8"?>
2 <ArrayOfDaqSetting xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3   <DaqSetting Name="ApplyZ2" SamplingRate="44100" BitPerSample="16" BufferLength_Millisecond="100">
4     <Channel Index="0" Name="Microphone" Type="AI" Quantity="SoundPressure" Sensitivity="100" />
5     <Channel Index="1" Name="Accelerometer" Type="AI" Quantity="Acceleration" Sensitivity="0.1" />
6     <Specification Device="ITRIAPPLYZ2" ChannelCount="2" MaxSamplingRate="48000" MaxBitPerSample="16" SignalMax="5"
7       SignalMin="-5" />
8   </DaqSetting>
9   <DaqSetting Name="VimoNet" SamplingRate="102400" BitPerSample="16" BufferLength_Millisecond="10240" IP="192.168.1.117">
10    <Channel Index="0" Name="Accelerometer" Type="AI" Quantity="Acceleration" Sensitivity="0.1" />
11    <Specification Device="VimoNet" ChannelCount="1" MaxSamplingRate="102400" MaxBitPerSample="24" SignalMax="5"
12      SignalMin="-5" />
13  </DaqSetting>
14 </ArrayOfDaqSetting>

```

亦可於 VMX 的設定頁面中點選 DAQ 標籤檢視 DAQ 設定與狀態：

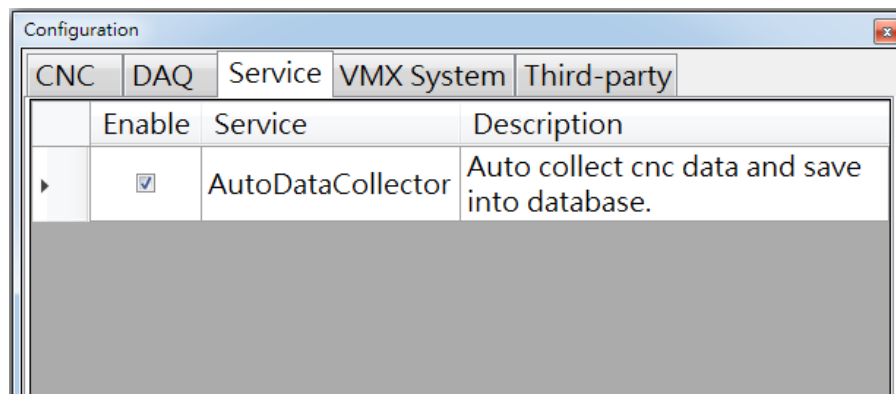


點選右上角的  可開啟 daq.xml 的檔案位置(預設路徑 Documents\ITRI VMX\Settings)、點選  圖示可檢視 DAQ 連線設定內容(目前只能透過直接編輯 XML 檔案更改連線設定)、點選  圖示可重新整理 DAQ 狀態、點選  /  可啟用/停用 DAQ、點選  可檢視當前的時域訊號：



2.2.3 服務設定

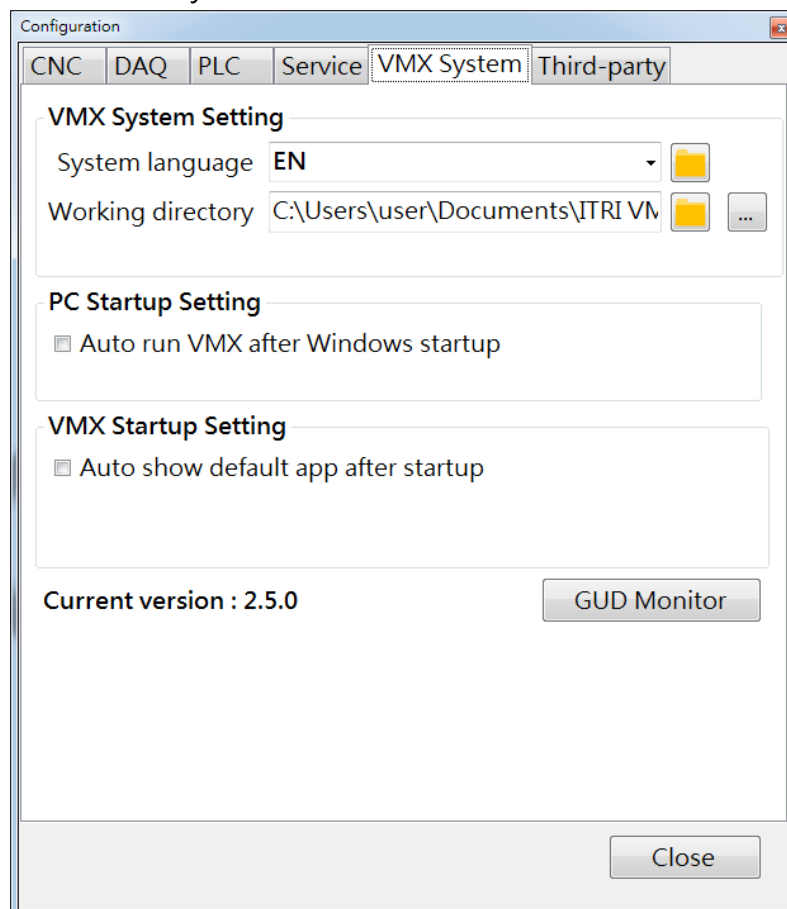
於 VMX 的設定頁面中點選 Service 標籤可檢視服務的設定與狀態：



勾選/取消勾選 Enable 可啟用/停用服務。

2.2.4 VMX 系統設定

於 VMX 的設定頁面中點選 VMX System 標籤可檢視 VMX 系統的的設定：



於 VMX System Setting 區塊中可更改 VMX 的語言和工作目錄。目前預設支援中文和英文，使用者可依照語言檔(例：vmxlang.ZHTW.txt)自行編輯擴充，最多可支援八國語言(中、英、日、韓、德、法、義、西)。工作目錄則為 VMX 讀取設定檔和寫入記錄檔的位置，預設為 MyDocuments\ITRI VMX。

於 PC Startup Setting 區塊中可透過勾選/取消勾選決定是否要在 Windows 啟動後自動執行 VMX。

於 VMX Startup Setting 區塊中可透過勾選/取消勾選決定是否要在 VMX 啟動時自動進入預設 APP 畫面。

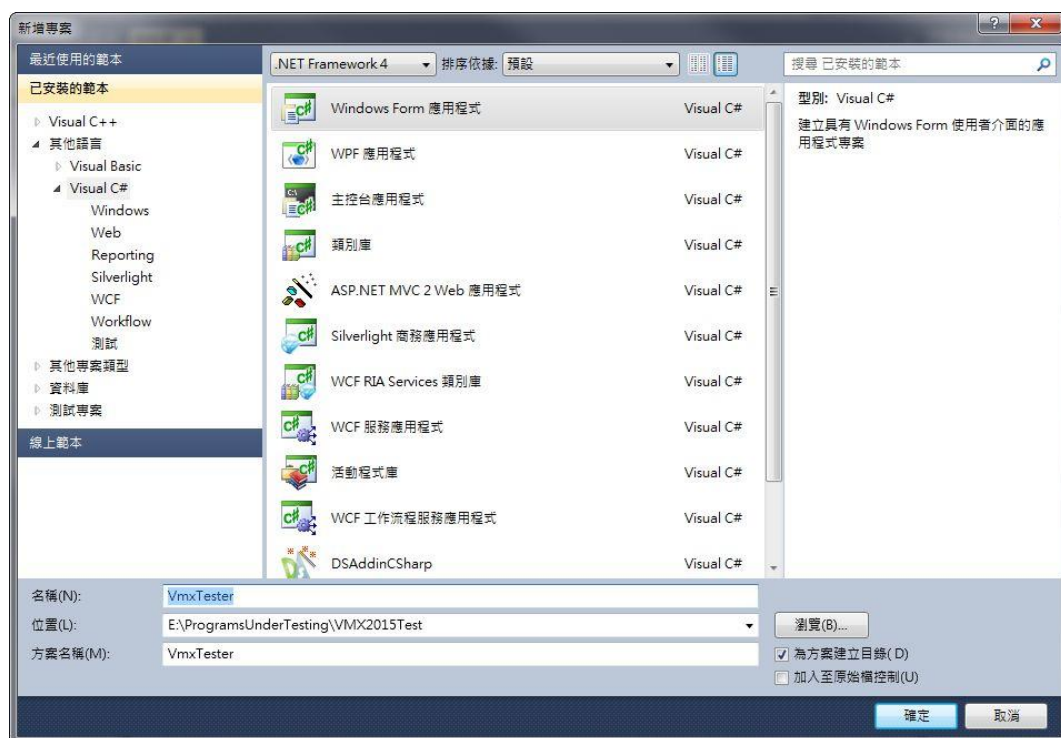
第三章、軟體開發

3.1 APP 掛載機制

為了讓使用者便捷開發智能軟體，VMX 選擇用 Windows Form 作為 APP 開發的主要環境，同時也支援以 WPF User Control 開發的 APP。

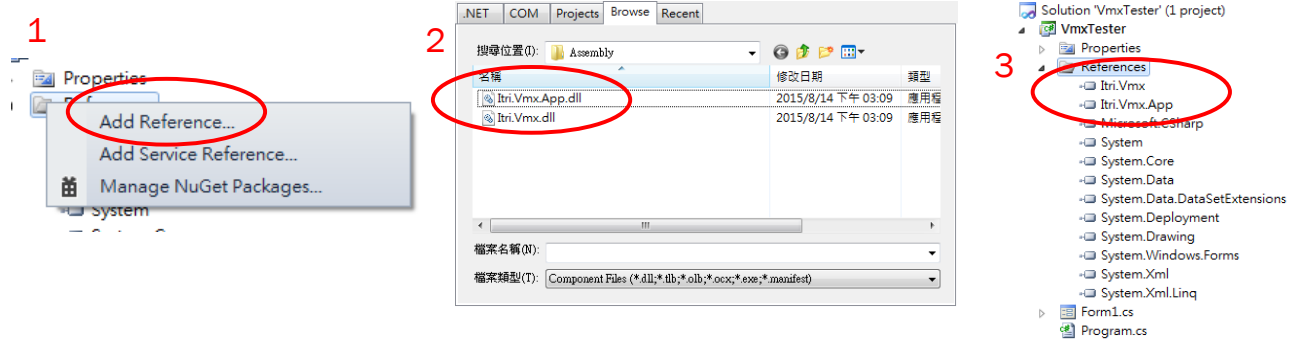
步驟 1、新增 Windows Form 專案

使用 Windows Form 作為 APP 開發環境，一開始的步驟與一般 Windows Form 應用程式的開頭步驟是一樣的：新增一個 Windows Form 專案。目前 VMX 可支援 .NET Framework 4.5.2、開發者可自由選擇相同或以下的 .NET 版本。



步驟 2、加入相關參考組件

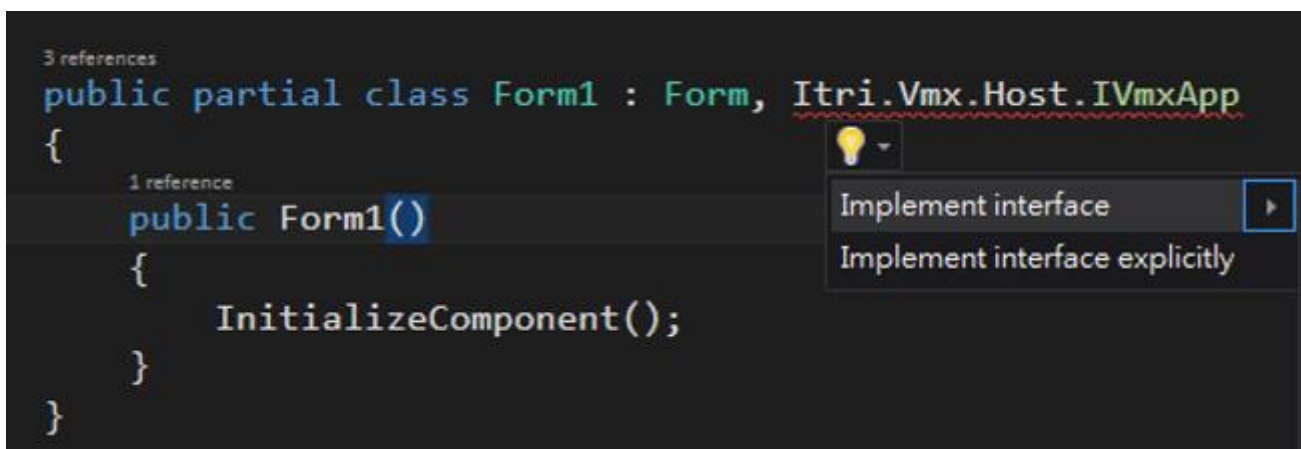
專案要加入相關的參考組件(Assembly)。在 .NET Framework 中，動態連結函式庫(Dynamic Link Libraries, DLL)是屬於一種組件。可以在方案管理中，增加組件的參考。在本例中，請於 VMX 的安裝資料夾中找到 BIN 資料夾，參考裡面的 Itri.Vmx.DLL 與 Itri.Vmx.App.DLL。完成後，在組件的參考清單中可以看到參考的兩個 DLL 檔案。



步驟 3、實作 APP 的介面

APP 要掛載入 VMX，需要實作 APP 對 VMX 的介面。VMX 利用微軟.NET Framework 4.0 的 Managed Extensibility Framework (MEF) 開發 APP 的載入架構，所以 APP 與 VMX 間存在一個 MEF 合約(Contract)。此合約定義在 Itri.Vmx.App.DLL 內，定義為.NET Framework 的一種介面 interface，APP 需要實作此一介面方能掛載入 VMX。

APP 與 VMX 間的合約 IVmxApp 的介面定義於 Itri.Vmx.Host 的命名空間(namespace)下。開發者在原本的 Form1(預設名稱，使用者可以改變)上繼承 IVmxApp，語法如下圖標示 2 處。接著在 IVmxApp 的文字處，選擇實作介面(Implement Interface)。



微軟的 Visual Studio 編譯器提供實作介面程式碼產出。但實作介面有兩種形式，一種是實作介面(Implement Interface)，即在物件(在此就是 Form1)內產生介面需要的方法(或稱為函式)；另一種是明確實作介面(Implement Interface Explicitly)是只有實作介面的方法，即僅限於物件轉型成介面實作之後才能使用的方法。無論何種形式 VMX 均能夠載入，但建議若原本 Form1 有同名方法時或是要明確區分 Form1 方法與合約方法時使用「明確實作介面」。



實作介面會產生 APP 需要實作的方法，應如下所列，包含 AppName、Image 與 Initialize 等方法。三個方法的用途為：

- AppName：APP 的顯示名稱。
- Image：APP 的顯示圖示。
- Initialize：APP 起始化的動作。通常用於檢查資源與圖形介面配置。

```
public string AppName
{
    get
    {
        return "SampleAPP";
    }
}

public Image Image
{
    get
    {
        return Image.FromFile("icon.jpg");
    }
}

public bool Initialize(IVmxHost host)
{
    return true;
}
```

步驟 4、匯出成 DLL 檔案

完成合約後，依據以下步驟將此專案匯出 DLL 檔即可掛載入 VMX 的主控制台(Host)內：

- 1 在專案的參考中增加 System.ComponentModel.Composition 組件。
- 2 在程式碼內加入命名空間使用(using namespace)和輸出類型(Export type)。

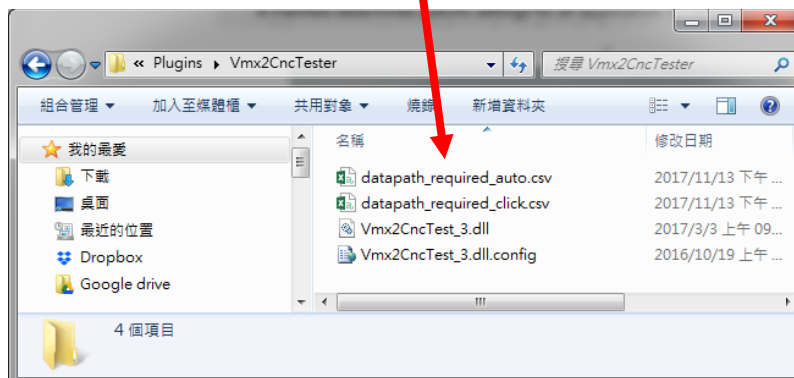
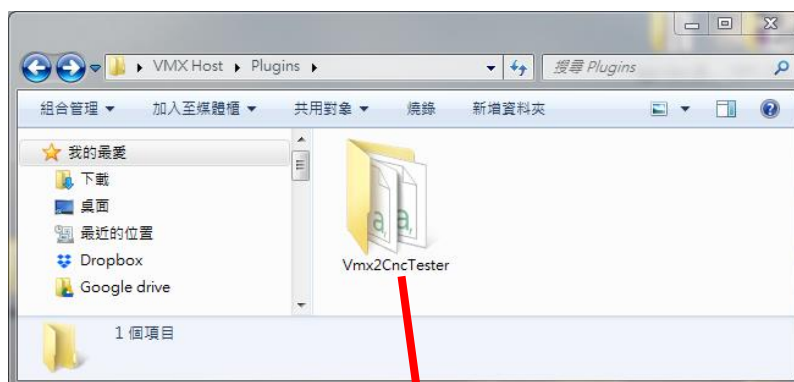
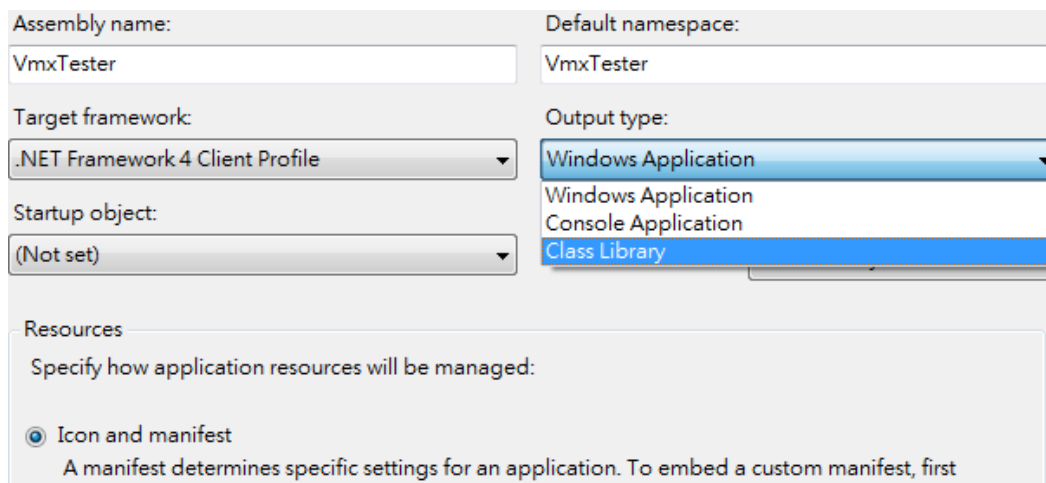
```
using Itri.Vmx.Host;
using System.ComponentModel.Composition;

namespace SampleApp
{
    [Export(typeof(IVmxApp))]
    public partial class Form1 : Form, IVmxApp
```



- 3 打開專案的屬性頁面。
- 4 在應用程式頁面上，將專案輸出類型改成類別函式庫 (Class Library)。
- 5 在建置的頁面上，將平台輸出改成 x64 或 x86 (根據 VMX 的版本平台決定)
- 6 建置專案。
- 7 將輸出的 DLL 檔置入 VMX 安裝目錄\Plugins 內。

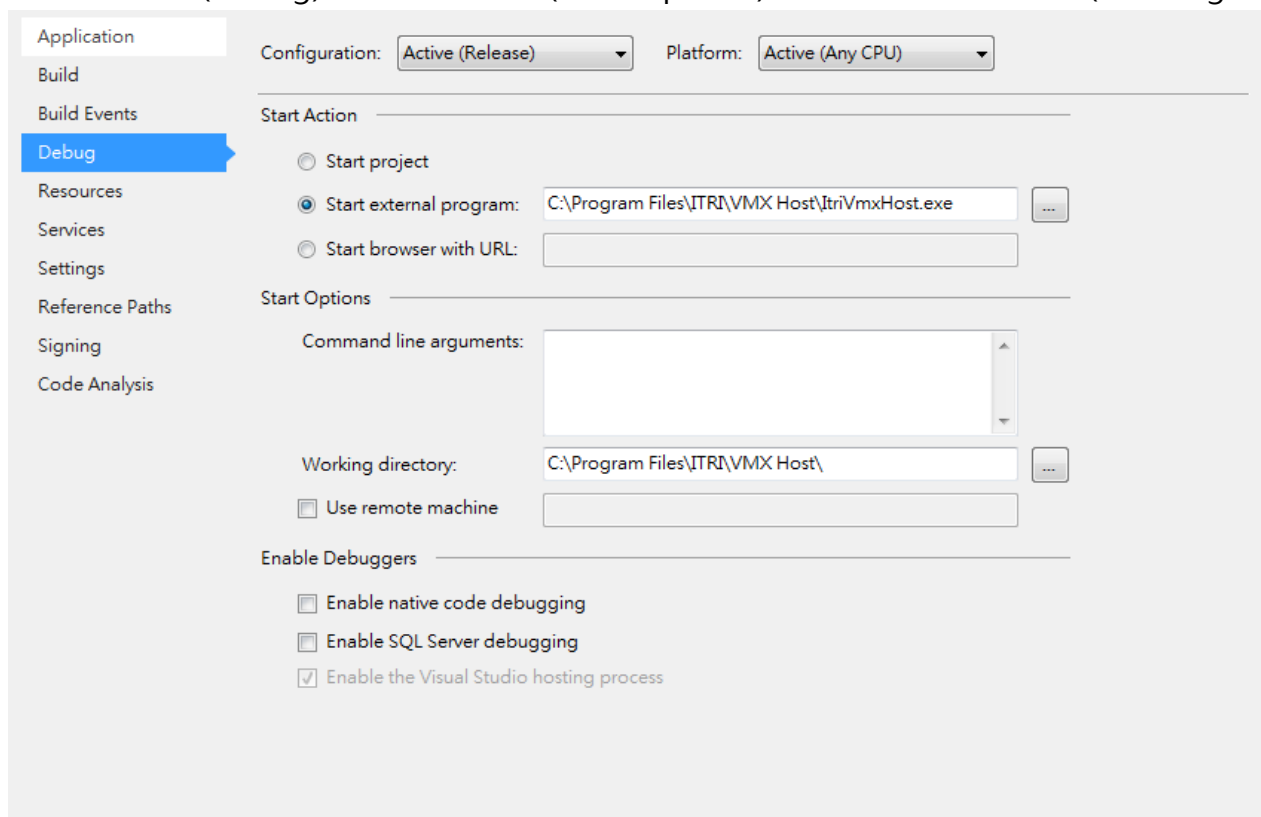
注意！每個 APP 應有獨立資料夾。



3.2 APP 的除錯環境設定

若使用 Visual Studio Professional 或更高階的版本，可以設定相關環境，即可使用 Visual Studio 的除錯機制進行 APP 的開發，可以便捷 APP 的開發。請開啟專案屬性頁面：

1. 在除錯(Debug)分頁的啟動動作(Start Action)群組中，啟用啟動外部程式(Start external program)，並選擇 VMX 執行檔。
2. 在除錯(Debug)分頁的啟動選項(Start Options)群組中，指定工作目錄(Working directory)。



The screenshot shows the 'Debug' tab in the Visual Studio 'Properties' window. The left sidebar lists various categories, with 'Debug' selected. The main area is divided into sections: 'Configuration' (Active (Release)), 'Platform' (Active (Any CPU)), 'Start Action', 'Start Options', and 'Enable Debuggers'. In the 'Start Action' section, 'Start external program' is selected, and the path 'C:\Program Files\ITRI\VMX Host\ItriVmxHost.exe' is entered. In the 'Start Options' section, the 'Working directory' is set to 'C:\Program Files\ITRI\VMX Host\'. Under 'Enable Debuggers', 'Enable the Visual Studio hosting process' is checked, while 'Enable native code debugging' and 'Enable SQL Server debugging' are unchecked.

Application

Build

Build Events

Debug

Resources

Services

Settings

Reference Paths

Signing

Code Analysis

Configuration: Active (Release) Platform: Active (Any CPU)

Start Action

☐ Start project

☒ Start external program: C:\Program Files\ITRI\VMX Host\ItriVmxHost.exe

☐ Start browser with URL:

Start Options

Command line arguments:

Working directory: C:\Program Files\ITRI\VMX Host\

☐ Use remote machine

Enable Debuggers

☐ Enable native code debugging

☐ Enable SQL Server debugging

☒ Enable the Visual Studio hosting process



3. 在建置(Build)分頁的輸出(Output)群組，指定專案輸出路徑(Output path)至 VMX 安裝目錄的\plugins 資料夾內。由於每個 APP 都需要獨立資料夾，所以務必增加一層目錄，例如：[VMX 安裝目錄]\plugins\[APP 目錄]。

The screenshot shows the Visual Studio Build configuration window. The left sidebar has 'Build' selected. The main area is divided into sections: 'General', 'Errors and warnings', 'Treat warnings as errors', and 'Output'. In the 'Output' section, the 'Output path' is set to 'C:\Program Files\ITRI\VMX Host\Plugins\vimopro2\'. There are also checkboxes for 'XML documentation file' and 'Register for COM interop', and a dropdown for 'Generate serialization assembly' set to 'Auto'. The 'Warning level' is set to '4'.

透過此一機制，Visual Studio 的除錯器會在指定的工作目錄啟動 VMX 主控台程式(ItriVmxHost.exe)，而 VMX 啟動的同時會自動載入開發中 APP，則除錯器就可以進程式碼的偵錯。

3.3 CNC 轉接器的應用

3.3.1 基本操作

I. 掛載通訊介面

依據各通訊介面的需求，將檔案置入 VMX 安裝目錄或 API 資料夾中，並透過 VMX 的設定 (Configuration) 建立和管理 CNC 連線設定，詳情可參考 2.2.1 的內容。

II. 於程式中取得 CNC 轉接器

(1) 參考 3.1 節內容，掛載 APP 於 VMX 中，並新增參考 IVmx2Cnc.dll

(2) 建立一 Itri.Vmx.Cnc.CncAdaptor 實體

```
Itri.Vmx.Cnc.CncAdaptor cnc;
```

(3) 於 VMX 合約之 Initialize() 方法中，取得 CNC 轉接器

```
cnc = host.CncAdaptors[0];
```

取得 CncAdaptors 的參數有兩種：

- Value index：如 CncAdaptors[0] 表示 VMX 中的第一個 CNC 連線，依照 CNC 機台名稱排序。
- String index：如 CncAdaptors["Fanuc01"] 表示想要取得機台名稱為 Fanuc01 的 CNC 連線，若沒有符合需求的目標則會回傳 null。

III. 透過通訊介面取得機台資料

(1) 建立一 Itri.Vmx.Cnc.DataItem 實例(Instance)

```
Itri.Vmx.Cnc.DataItem item = new Itri.Vmx.Cnc.DataItem();
```

(2) 設定資料路徑 (例：進給速度)

```
item.Path = "/Controller/Feedrate" ;
```

其中路徑可參考附錄提供之 CNC 資料路徑表

(3) 呼叫 ReadDataItem()方法

```
int errorCode = cnc.ReadDataItem(ref item);
```

由於此處使用 ref 傳址的方式，取得的資料會填回 item 的 Value 屬性中，而 ReadDataItem 會回傳一 int 數值，表示是否成功取回資料，或失敗時錯誤的原因，對應可參考附錄提供之 CNC 錯誤代碼表

(4) 取出數值並轉型

```
double feedrate = Convert.ToDouble(item.Value);
```

其中轉型可利用 Convert 類別提供之方法如：ToDouble、ToInt32、ToString 等

3.3.2 程式碼範例

```
Itri.Vmx.Cnc.CncAdaptor cnc = null;
public bool Initialize(Itri.Vmx.Host.IVmxHost host)
{
    if (host.CncAdapters.Length != 0)
    {
        cnc = host.CncAdapters[0];
        cnc.KeepAlive = true;
    }
    return true;
}

Itri.Vmx.Cnc.DataItem feedrate = new Itri.Vmx.Cnc.DataItem();
Public double GetFeedrate()
{
    feedrate.Path = "/Controller/Feedrate";
    cnc.ReadDataItem(ref feedrate);
    return Convert.ToDouble(feedrate.Value);
}
```


3.4 DAQ 轉接器的應用

3.4.1 設定 DAQ 轉接器

VMX 提供 XML 檔案，可以設定一個或更多的 DAQ 連接器，設定檔如下，各種 DAQ 的設定根據廠牌有所不同。

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfDaqSetting xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <DaqSetting Name="DefaultName" SamplingRate="12800" BitPerSample="16"
  BufferLength_Millisecond="500">
    <Channel Index="0" Name="Microphone1" Type="AI" Quantity="SoundPressure" Sensitivity="100" />
    <Specification Device="ITRIAPPLYZ2" ChannelCount="2" MaxSamplingRate="48000" MaxBitPerSample="16"
    SignalMax="5" SignalMin="-5" />
  </DaqSetting>
</ArrayOfDaqSetting>
```

當 VMX 主控台啟動後，會自動根據設定檔建立控制器連接器資源。VMX 主控台透過 VMX 合約的 Initialize() 傳遞各式資源給加值軟體。

```
private Itri.Vmx.Daq.DaqAdaptor daq = null;

public bool Initialize(Itri.Vmx.Host.IVmxHost host)
{
    if(host.DaqAdaptors != null && host.DaqAdaptors.Length > 1)
        daq = host.DaqAdaptors[0];
    return daq == null ? false : true;
}
```

3.4.2 使用 DAQ 轉接器

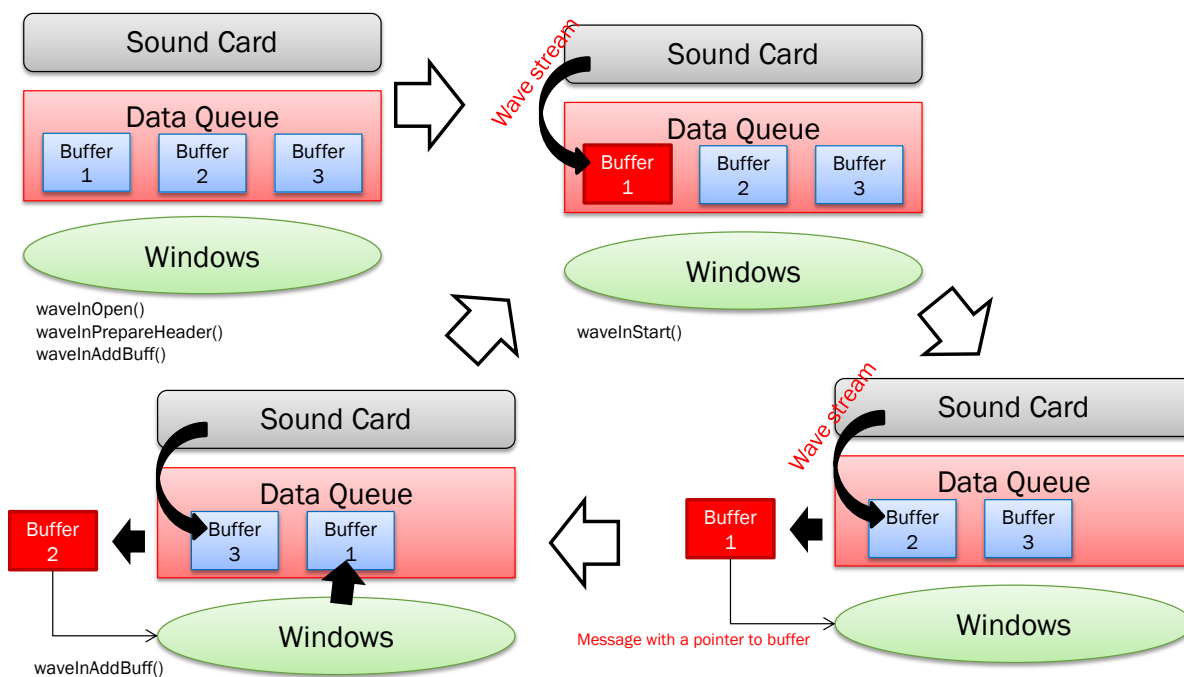
以下說明 DaqAdapter 使用方式：

1. 編輯 XML 設定 DAQ，並啟動 VMX。
2. 訂閱 DataAcquired 事件。DaqAdapter 會以事件的方式發送時序訊號。
3. 啟動 DaqAdapter。啟動後，DaqAdapter 會根據設定頻率發送 DataAcquired 事件。(發送出的訊號均有時間戳記)
4. 停止 DaqAdapter。停止時，DaqAdapter 會停止發送 DataAcquired 事件。

由於直接使用 DaqAdapter 會有衝突或資源問題，VMX 團隊推薦 3.4.4 節的操作方法。

3.4.3 DAQ 設備的運作概要

一般 DAQ 設備的 API 操作有二種類型。第一種是多緩衝區型，如下圖所示的視窗多媒體函數庫 (windows multimedia library, winmm.dll) 操作方式，通常操作的硬體是電腦的音效卡。首先開啟硬體、準備緩衝區且置入資料佇列內，一般至少要有二個緩衝，避免資料遺漏。接著啟動資料擷取功能 (此處為 waveInStart)，將訊號根據擷取的格式收入緩衝區內。當緩衝區收滿時，會從資料佇列退出，同時下一個緩衝區會自動補位，新的資料會繼續存入補位的緩衝區內。緩衝區的退出通常會附帶一個事件(event)或回呼(callback)機制；若開發者利用事件或回呼機制，則資料會直接傳遞到開發者準備好的工作流程，例如進行訊號處理。當緩衝區被使用後，應重設該緩衝區並再次加入資料佇列內。透過此一循環，可以確保資料不斷的收集。多緩衝區類型的操作方式允許多種應用程式同時訂閱設備的事件或回呼機制，但效能較差；也因為事件或回呼攜帶的指標(pointer)或參考(reference)指向的是同一個緩衝區記憶體，資料的安全性較有疑慮。



另一種操作方式是單緩衝區類型，也就是只有一個緩衝區，硬體收集的訊號直接存去該緩衝區內，而開發者取資料時是直接取走所有已存好的資料。美國國家儀器(National Instruments, NI)的設備與軟體是屬於此類型的操作。此類型的優點是確保開發者每次取得的資料都是最新且獨有，效能也比較高。但資料被取走後，原來的資料就不在緩衝區內，其他應用程式無法再取得該資料 (實際上也不允許這麼做)。因為設備獨佔的緣故，一個感測器只能被一個應用程式使用(access)；如果要多種應用程式取得相同的資料，則不外乎硬體數量增加，或增加跨程序的軟體整合工作，都造成硬體或人力上的成本增加。

3.4.4 使用 DAQ 緩衝

VMX 對多 APP 存取 DAQ 資源的解決方案是：

1. 由 VMX 獨佔 DAQ 的資源並且轉散發至 APP。
2. **VMX 提供一個 DAQ 資源操作類別 DaqBuffer**，產生一個仿 DAQ 的暫存區，VMX 會自動將擷取的信號直接發送到該暫存區。
 - 注意！VMX 的機制會造成即時性略差。需高度即時性的 APP 目前建議採用自有的 DAQ 介面。

DaqBuffer 的操作如下：

a. 產生 DAQ 緩衝器實例/Create DaqBuffer instances

```
Itri.Vmx.DaqBuffer buffer  
= new DaqBuffer(  
    host.DaqAdaptors[0],  
    host.DaqAdaptors[0].Settings.SamplingRate * 1);
```

b. 產生 APP 內的容器 (以單通道為例)

- a) 取出全部資料模式：呼叫緩衝器的容器創造方法。

```
double[] allData = buffer.CreateContainerForBufferData();
```

- b) 取出新資料模式：不需要處理

```
double[] newData = null;
```

c. 取得全部訊號資料

- a) 直接取用模式/Check mode：直接呼叫 GetAllData() 方法取出資料。

```
buffer.GetAllData(channel, allData);  
//your signal process...
```



b) 事件模式/Event mode

1. 建立事件處理方法

```
private void GetEventData(object s, EventArgs e)
{
    if (s is Itri.Vmx.DaqBuffer)
    {
        buffer.GetAllData(channel, allData);
        //your signal process...
    }
}
```

2. 訂閱事件與啟用事件

```
buffer.DataReady += this.GetEventData;
buffer.EnableDataReady = true;
```

d. 取得新的訊號資料

- a) 直接取用模式/Check mode：透過 IsUpdated 檢查是否更新後，呼叫 GetNewData()方法取出資料。

```
if(buffer.IsUpdated)
    buffer.GetNewData(channel, out newData);
//your signal process...
```

b) 事件模式/Event mode

1. 建立事件處理方法

```
private void GetEventData(object s, EventArgs e)
{
    if (s is Itri.Vmx.DaqBuffer)
    {
        buffer.GetNewData(channel, out newData);
        //your signal process...
    }
}
```

2. 訂閱事件與啟用事件

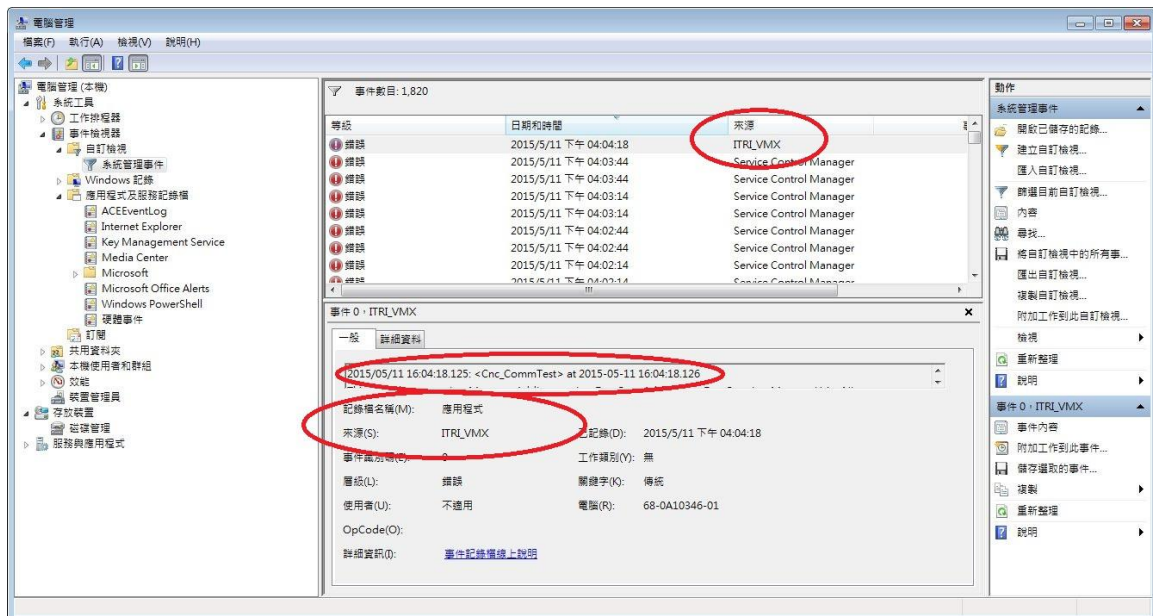
```
buffer.DataReady += this.GetEventData;
buffer.EnableDataReady = true;
```

3.5 歷程記錄器的應用

CommonLogger 是基於第三方套件 NLog，可由 NLog.config 檔修改輸出形式。CommonLogger 提供六種程度的紀錄手法，包含 Trace, Debug, Warn, Error, Info, Fatal 等。由於 CommonLogger 是一個靜態類別，使用時不需建立任何實例，直接調用靜態函式使用。以下是程式碼的範例。

```
CommonLogger.Trace("MyAppName", "This is trace message");  
CommonLogger.Debug("MyAppName", "This is debug message");  
CommonLogger.Info("MyAppName", "This is info message");  
CommonLogger.Warn("MyAppName", "This is warn message");  
CommonLogger.Error("MyAppName", "This is error message");  
CommonLogger.Fatal("MyAppName", "This is Fatal message");
```

根據記錄的方式，記錄訊息會儲存在不同的檔案或系統服務上。多數的歷程記錄都會存在 VMX 安裝資料夾內的 log 資料夾內。但嚴重錯誤(Fatal)則會特別儲存在視窗(Windows)作業系統的系統日誌內。VMX 具有異常攔截機制，可以避免 APP 異常發生時造成 VMX 當機或終止，並將嚴重錯誤記錄下來。但若錯誤來自於系統性的缺失，例如記憶體操作錯誤，則 VMX 有可能無法攔截異常及儲存異常的相關記錄。



VMX 團隊建議開發者利用歷程記錄器的機制，建立軟體錯誤追蹤機制，以利於軟體未來的除錯與更新。

3.6 APP 間的溝通機制

VMX 團隊注意到應用程式間的溝通問題(issue)，是智能機械軟體整合的另一道障礙。過去智能應用軟體多是各自為政，每個軟體都是獨立的程序(Process)且程序間的溝通管道有限；若非要溝通，則通常要透過其他方式處理，例如檔案交換、Socket 傳輸方式或記憶體共享。然而，當智能軟體作為 APP 掛載入 VMX 時，VMX 提供了廣播與訂閱機制，輕鬆解決 APP 間的溝通問題。以下是 APP 之間溝通的範例：

- 訊息發送的 APP，其 AppName 為 SenderApp：

1. 在 Initialize 中取得 host 資源。

```
Itri.Vmx.Host.IVmxHost _host = null;

public bool Initialize(Itri.Vmx.Host.IVmxHost host)
{
    _host = host;
    return true;
}
```

2. 透過 host 的 SendMessage 方法，將訊息發送出去。訊息是透過 VmxMessage 發送，可以附加物件。

```
private void SendMessageToVmxHost()
{
    Itri.Vmx.VmxMessage msg = new Itri.Vmx.VmxMessage();
    msg.Tag = "Hello, World";
    msg.AttachedObject = new string[] { "Jason", "183" };
    _host.SendMessage(this,msg);
}
```

- 接受訊息的 APP：

1. 在 Initialize 中，訂閱 host 的 MessageArrived 事件。

```
public bool Initialize(Itri.Vmx.Host.IVmxHost host)
{
    host.MessageArrived += this.GetMessageFromVmxHost;
    return true;
}
```



2. 實作訊息接受介面。其中解開附加物件的方法，可以雙方 APP 約定，或是用 C# 的 var 型別拆解。

```
private void GetMessageFromVmxHost(object s, Itri.Vmx.VmxMessage msg )
{
    if((s as Itri.Vmx.Host.IVmxApp).AppName == "SenderApp")
    {
        string[] package = (string[]) msg.AttachedObject;
        string combinedMsg = string.Format("{0}'s height is {1} cm.");
    }
}
```

第四章、可自製的通訊/操作介面

VMX 提供開發者自行發展通訊介面。若開發者使用的資訊模型(information model)與 VMX 的標準一致 (參見附件：CNC 資料路徑表)，則開發者所完成的 APP 或服務就能通用在其他類似的設備上。此種開發方式有利於採用自製設備或擁有的設備類型有限的學研單位，可確保研發的軟體產品也能應用在其他商用設備上。以下說明 CNC 通訊介面的自製方法，其他智慧機械或設備 (例如 PLC 或機器人) 的作法幾乎一致。DAQ 或感測器操作介面部分正在研擬中(版本 2.3.0)，未來將會推出可自製化的 DAQ 與感測器操作介面。

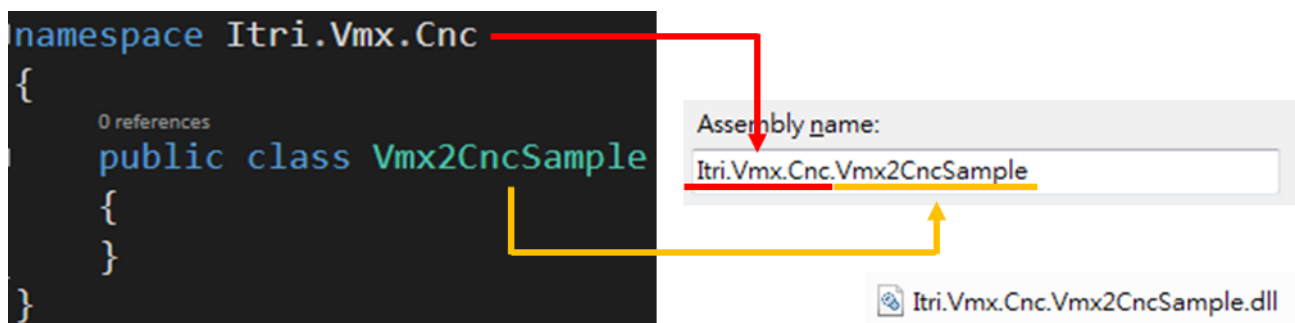
注意！第四章內容需要熟習 API 的開發過程，建議新手開發者可洽詢 VMX 團隊的工程師。

4.1 CNC 通訊介面

VMX 的通訊介面建議使用 C#作為開發語言，.NET 版本允許最低 4.0、最高 4.5.2，輸出類型為類別庫 (Class Library)，即 DLL 檔案。

為了使用 VMX 中的各項資源與定義，開發者需將 IVmx2Cnc.DLL 加入專案參考中，該檔案可在 VMX 的安裝目錄中找到 (預設路徑：C:\Program Files\ITRI\VMX Host)。

為了使 VMX 可正確載入所需的組件及類別，通訊介面自身的「介面類別」所在之 DLL 檔檔名必須為「介面類別」的命名空間加上類別名稱，範例如下：





4.1.1 介面繼承

由 VMX 所定義的 CNC 控制器標準化通訊介面定義於 Itri.Vmx.Cnc.IVmx2Cnc 中，通訊介面的開發者需將自身的介面類別繼承及實作該介面，範例如下：

```
public class Vmx2CncSample : IVmx2Cnc
{
    public CncDataCollection CncDataBuffer
    {
        get { throw new NotImplementedException(); }
    }
    public ConnectingSettingCollection ConnectingSetting
    {
        get { throw new NotImplementedException(); }
        set { throw new NotImplementedException(); }
    }
    public bool IsConnected
    {
        get { throw new NotImplementedException(); }
    }
    public int Connect()
    {
        throw new NotImplementedException();
    }
    public int Disconnect()
    {
        throw new NotImplementedException();
    }
    public int ReadDataItem(ref DataItem result)
    {
        throw new NotImplementedException();
    }
    public int WriteDataItem(DataItem data)
    {
        throw new NotImplementedException();
    }
    public void Dispose()
    {
    }
}
```

4.1.2 介面定義與實作方法

● CncDataCollection CncDataBuffer

CNC 資料暫存空間。通訊介面必須具有一自動且持續收集資料的機制，並將取得的資料預先放入此空間中。此空間可供 VMX 無延遲得讀取資料，並提供資料變更時的訂閱機制。通訊介面應宣告並建構一私有的 CncDataCollection 類型變數，並於此屬性中提供，如：

```
int[] idList = { 60, 61, 70, 2000, 2010 };
CncDataCollection buffer;

public Vmx2CncSample()
{
    buffer = new CncDataCollection(idList);
}

public CncDataCollection CncDataBuffer
{
    get
    {
        return buffer;
    }
}
```

其中 idList 包含此通訊介面所支援之路徑 ID (路徑與 ID 的對照可參考 VMX 使用手冊的附錄：IVmx2Cnc DataItem Manual)。

由於此暫存空間會對 APP 直接開放，為了保護資料的正確性，此暫存空間無法直接對其中的特定元素寫入資料，需透過轉型呼叫特定方法，範例如下：

```
(buffer[70] as ICncDataSet).SetValue(0, 1000.0);
```

在建構 CncDataCollection 時，會根據傳入的 idList 建立相對應的「格子」，故只有 idList 中存在的 ID 才被允許用於取得特定的資料暫存空間。SetValue 需求兩個參數：

- **int errorCode**：錯誤代碼，標示此作業成功與否
- **object value**：資料內容，實際需填入的資料型別視不同資料而定，可參考 VMX 使用手冊的附錄：IVmx2Cnc DataItem Manual)

- **ConnectingSettingCollection** ConnectingSetting

連線設定。VMX 載入 XML 設定檔後，會將所有資訊於此傳遞給通訊介面。通訊介面應宣告並建構一私有的 CncDataCollection 類型變數，並於此屬性中提供讀取與寫入，如：

```
ConnectingSettingCollection setting;

public Vmx2CncSample()
{
    setting = new ConnectingSettingCollection(new ConnectingSettingItem[]
    {
        new ConnectingSettingItem("IP", "127.0.0.1"),
        new ConnectingSettingItem("Port", "12345")
    });
}

public ConnectingSettingCollection ConnectingSetting
{
    get
    {
        return setting;
    }

    set
    {
        setting = value;
    }
}
```

其中該 ConnectingSettingCollection 變數建構時需傳入一 ConnectingSettingItem 陣列，每一個 ConnectingSettingItem 代表一項此通訊介面的連線資訊，ConnectingSettingItem 建構時需一併傳入此項連線資訊的名稱與預設值。當 VMX 建構此通訊介面後，會由此屬性將使用者編寫之 XML 設定檔資訊傳入，故通訊介面應保留傳入的設定內容，並用於建立相對應的控制器連線。

- **bool** IsConnected

CNC 連線狀態，通訊介面需回傳布林值，表示當前連線存活與否。

- **int** Connect()

呼叫連線，VMX 會由此方法要求通訊介面建立與控制器的連線，需回傳一整數，為錯誤代碼，標示此作業成功與否。

- `int Disconnect()`

呼叫離線。VMX 會由此方法要求通訊介面關閉與控制器的連線，需回傳一整數代表錯誤代碼，標示此作業成功與否。

- `int ReadDataItem(ref DataItem result)`

讀取資料。VMX 會由此方法要求通訊介面讀取控制器上特定的資料，其資料目標與數值會由 `DataItem` 類型的參數 `result` 傳入與傳出，需回傳一整數代表錯誤代碼，標示此作業成功與否。

由參數 `result` 中可得知需求資料目標，標示於 `result.Path` 屬性中（可參考 VMX 使用手冊的附錄：IVmx2Cnc DataItem Manual），每一個路徑皆對應一種資料與數值型態，通訊介面需由路徑判斷資料目標，並將取得的資料轉型為特定的型態再放入 `result.Value` 中，範例如下：

```
public int ReadDataItem(ref DataItem result)
{
    int id = DataItem.GetPathId(result.Path);
    int ec;

    switch (id)
    {
        case 70:
        {
            result.Value = 1000.0; //read real value from controller
            ec = 0;
            break;
        }
        default:
        {
            ec = -5; //error code = -5: not support this path
            break;
        }
    }

    return ec;
}
```

- `int WriteDataItem(DataItem data)`

寫入資料。VMX 會由此方法要求通訊介面寫入特定資料至控制器中，其資料目標與數值會由 `DataItem` 類型的參數 `data` 傳入，需回傳一整數代表錯誤代碼，標示此作業成功與否。

由參數 `data` 中可得知資料目標與數值，標示於 `data.Path` 和 `data.Value` 屬性中（可參考 VMX 使用手冊的附錄：IVmx2Cnc DataItem Manual），通訊介面需由路徑判斷資料目標，並將數值寫入控制器中，範例如下：

```
public int WriteDataItem(DataItem data)
{
    int id = DataItem.GetPathId(data.Path);
    int ec;


    switch (id)
    {
        case 70:
        {
            SetDataToController(data.Value); //write value to controller
            ec = 0;
            break;
        }
        default:
        {
            ec = -5; //error code = -5: not support this path
            break;
        }
    }

    return ec;
}
```

4.1.3 測試

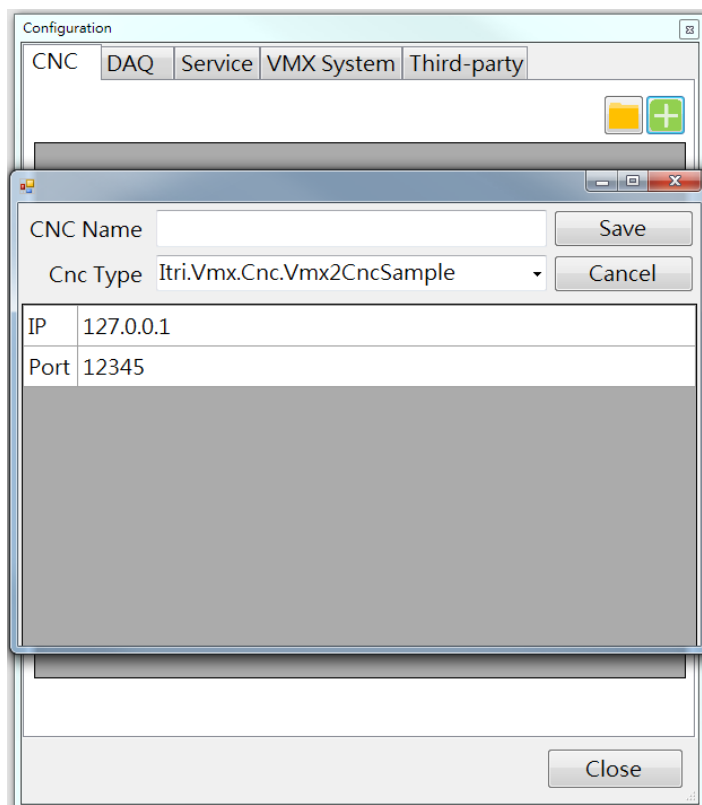
為了在 VMX 中使用該通訊介面，需將輸出之通訊介面檔案置入 VMX 安裝目錄中的 Api 資料夾中 (預設路徑為：C:\Program Files\ITRI\VMX Host\Api)，如果通訊介面有加入其他的參考檔案 (特指非.NET 的組件)，需將所有參考組件檔案一併放入 VMX 安裝目錄的根目錄中 (非 Api 資料夾)。



開啟 VMX 後，可由右上角的  圖示開啟設定頁面；於頁面上方選擇 CNC 分頁後，點選



可開啟新增機台連線設定頁面，此時應可於機台類型 (CNC Type) 下拉式選單中選擇自製的通訊介面，並於下方表格檢視與修改連線設定內容：

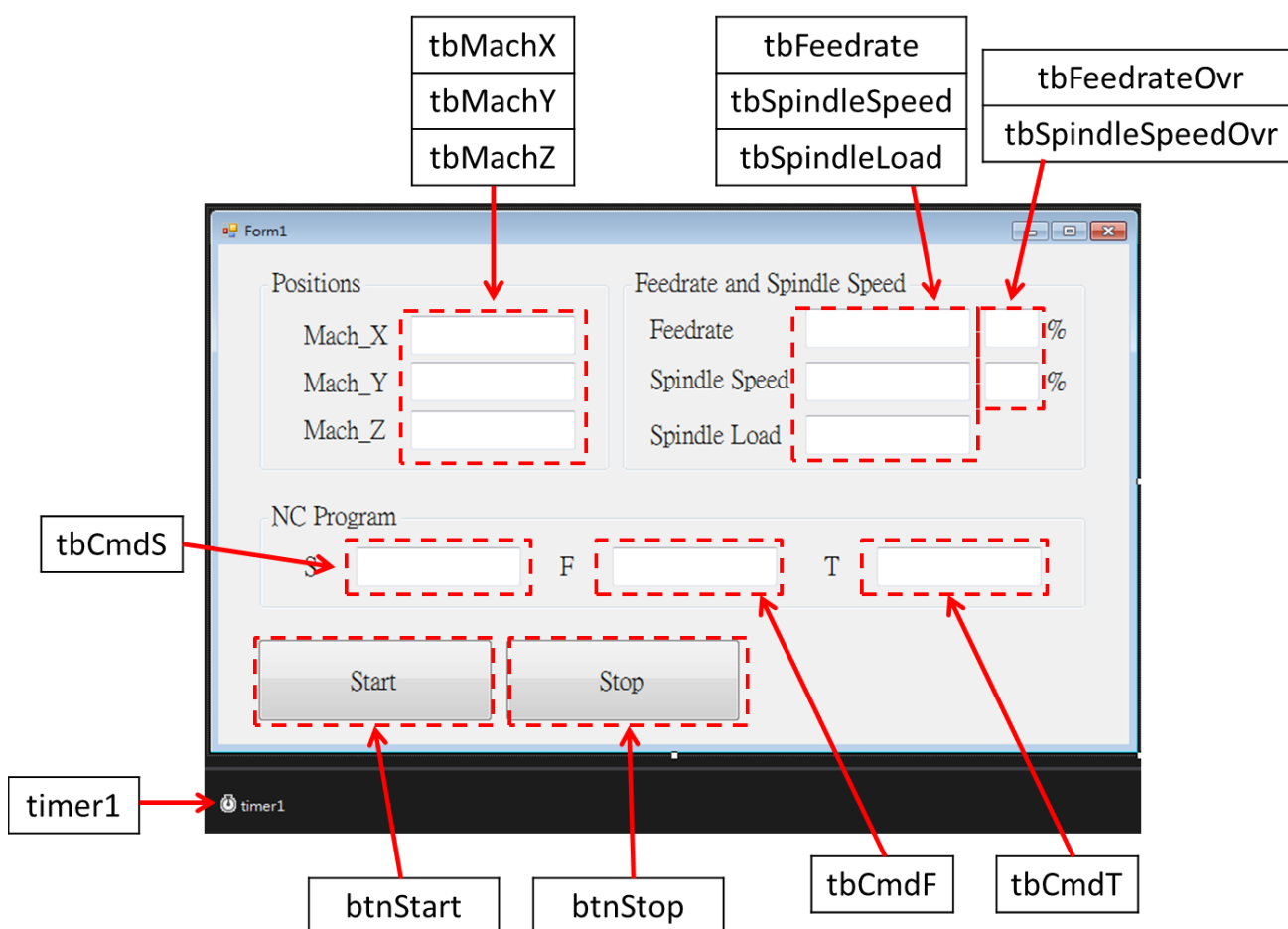


關於在 APP 中取得及使用通訊介面的方法，可參考 VMX 使用手冊。

第五章、應用案例

5.1 CNC 資料擷取範例

在此範例中，會從機台控制器取得三軸機械座標、進給速度、進給速度百分比、主軸轉速、主軸轉速百分比、主軸負載、轉速命令、進給命令、刀具命令等資訊呈現在 UI 上，並設置一計時器(timer)持續更新資料，此計時器可由按鈕控制開始和停止。以下為 UI 畫面，包含各控制項的命名，以及完整的程式碼。



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```



```
using System.Windows.Forms;

using System.ComponentModel.Composition;
using Itri.Vmx.Host;
using Itri.Vmx.Cnc;

namespace ex01_final
{
    [Export(typeof(IVmxApp))]
    public partial class Form1 : Form, IVmxApp
    {
        //1. Set VMX Constract
        public string AppName
        {
            get { return "ex01"; }
        }
        public Image Image
        {
            get { return Properties.Resources.ITRIVMX; }
        }
        CncAdaptor cnc = null;
        public bool Initialize(IVmxHost host)
        {
            if (host.CncAdapters.Length != 0)
            {
                cnc = host.CncAdapters[0];
            }

            return true;
        }

        //2. New DataItem Members
        DataItem mach_pos = new DataItem();
        DataItem feedrate = new DataItem();
        DataItem feedrateOvr = new DataItem();
        DataItem spindleSpeed = new DataItem();
        DataItem spindleOvr = new DataItem();
        DataItem spindleLoad = new DataItem();
        DataItem sCode = new DataItem();
        DataItem fCode = new DataItem();
    }
}
```




```
DataItem tCode = new DataItem();

public Form1()
{
    InitializeComponent();

    //3. Set DataItem Path
    mach_pos.Path = "/axes/MachineryPositions";
    feedrate.Path = "/controller/Feedrate";
    feedrateOvr.Path = "/controller/FeedrateOverride";
    spindleSpeed.Path = "/Spindle/ActualSpeed";
    spindleOvr.Path = "/Spindle/SpeedOverride";
    spindleLoad.Path = "/Spindle/Load";
    sCode.Path = "/controller/SpindleSpeedCmd";
    fCode.Path = "/controller/feedrateCmd";
    tCode.Path = "/controller/ToolIdCmd";
}

private void btnStart_Click(object sender, EventArgs e)
{
    timer1.Start();
}

private void btnStop_Click(object sender, EventArgs e)
{
    timer1.Stop();
}

private void timer1_Tick(object sender, EventArgs e)
{
    timer1.Stop();

    //4. Read Value
    cnc.ReadDataItem(ref mach_pos);
    cnc.ReadDataItem(ref feedrate);
    cnc.ReadDataItem(ref feedrateOvr);
    cnc.ReadDataItem(ref spindleSpeed);
    cnc.ReadDataItem(ref spindleOvr);
    cnc.ReadDataItem(ref spindleLoad);
    cnc.ReadDataItem(ref sCode);
}
```

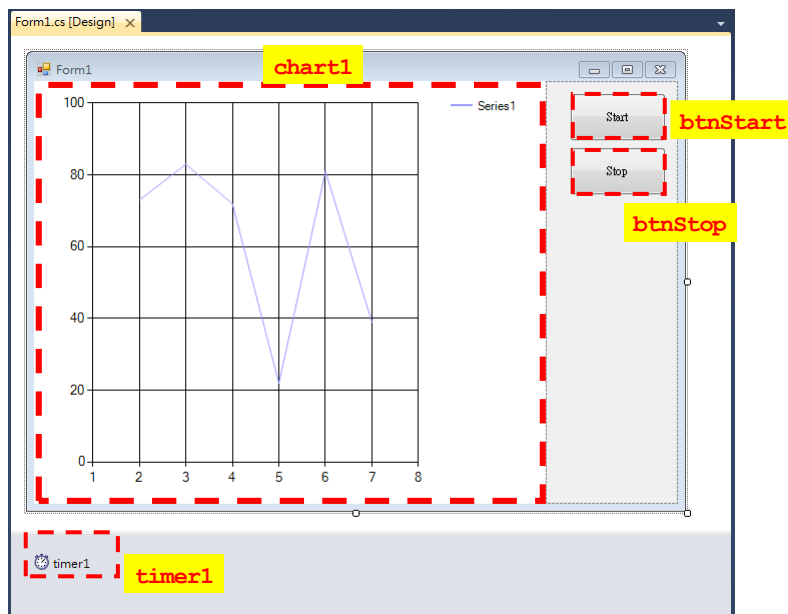


```
cnc.ReadDataItem(ref fCode);
cnc.ReadDataItem(ref tCode);

//5. Show Value
tbMachX.Text = (mach_pos.Value as Array).GetValue(0).ToString();
tbMachY.Text = (mach_pos.Value as Array).GetValue(1).ToString();
tbMachZ.Text = (mach_pos.Value as Array).GetValue(2).ToString();
tbFeedrate.Text = feedrate.Value.ToString();
tbFeedrateOvr.Text = feedrateOvr.Value.ToString();
tbSpindleSpeed.Text = spindleSpeed.Value.ToString();
tbSpindleSpeedOvr.Text = spindleOvr.Value.ToString();
tbSpindleLoad.Text = spindleLoad.Value.ToString();
tbCmdS.Text = sCode.Value.ToString();
tbCmdF.Text = fCode.Value.ToString();
tbCmdT.Text = tCode.Value.ToString();

timer1.Start();
    }
}
}
```

5.2 訊號擷取範例



```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.ComponentModel.Composition;
using Itri.Vmx.Host;
```

```
namespace ex04
```

```
{
    [Export(typeof(IVmxApp))]
    public partial class Form1 : Form, IVmxApp
```

```
{
    public Form1()
    {
        InitializeComponent();
    }
}
```

```
#region VMX contract
```

```
string IVmxApp.AppName
```

```
{
    get { return "Ex04_DaqApp"; }
}
```

```
Image IVmxApp.Image
```



```
{
    get { return Image.FromFile("icon.jpg"); }
}

Itri.Vmx.Daq.DaqAdaptor daq = null;
bool IVmxApp.Initialize(IVmxHost host)
{
    if (host != null)
    {
        //取得 Daq 轉接器
        if (host.DaqAdaptors[0] != null)
            daq = host.DaqAdaptors[0];

        //若 DAQ 尚未啟動，則啟動 Daq 轉接器
        if(daq.IsStarted == false)
            daq.Start();
    }
    if(daq == null)
        return false;
    return true;
}
#endregion

//資料成員
Itri.Vmx.Daq.DaqBuffer buffer = null;
double[] container = null;

private void btnStart_Click(object sender, EventArgs e)
{
    //更新 buffer 與儲存器
    if(buffer == null)
        buffer = new Itri.Vmx.Daq.DaqBuffer(daq, daq.Settings.SamplingRate*1);
    if (container == null)
        container = buffer.CreateContainerForBufferData();
    //啟動 timer 開始顯示資料
    timer1.Start();
}

private void btnStop_Click(object sender, EventArgs e)
{

```



```
//關閉 timer 停止顯示資料
timer1.Stop();
//釋放 buffer 與儲存器
container = null;
buffer = null;
}

private void timer1_Tick(object sender, EventArgs e)
{
    timer1.Stop();

    //從 buffer 取出所有的資料，存放在 container 內
    buffer.GetAllData(0, container);

    //將 container 顯示於畫面上
    chart1.Series[0].Points.DataBindY(container);

    timer1.Start();
}
}
```



IVmx2Cnc DataItem Manual

VMX 2.4.2

林錦德、王培寧、徐立宇
林怡瑾、黃琮琳、謝男凱

工業技術研究院 智慧機械科技中心
2018.11.16



目錄

50: /Controller/Guid.....	1
60: /Controller/OperatingMode.....	2
61: /Controller/OperatingStatus	3
62: /Controller/AlarmStatus	4
64: /Controller/RunTime.....	5
65: /Controller/OperationTime.....	6
66: /Controller/CycleTime.....	7
67: /Controller/PartCount.....	8
69: /Controller/RapidOverride.....	9
70: /Controller/FeedRate.....	10
71: /Controller/FeedRateOverride	11
72: /Controller/Program.....	12
73: /Controller/Line.....	13
80: /Controller/CurrentBlock.....	14
81: /Controller/PreReadBlocks.....	15
82: /Controller/SpindleSpeedCmd	16
83: /Controller/FeedRateCmd	17
84: /Controller/ToolIdCmd.....	18
85: /Controller/ToolLengthOffsetCmd.....	19
87: /Controller/GCodes	20
90: /Controller/Log/AlarmHistory	21
1800: /Compensation/ActiveWorkPieceOriginPointOffsetName	22
1801: /Compensation/WorkPieceOriginOffsetValue	23
1900: /Compensation/ToolOffsetFormat	24
1901: /Compensation/ToolOffsetTable.....	25
2000: /Axes/AxesCount	26
2001: /Axes/AxesNameList.....	27
2002: /Axes/AxesName.....	28
2005: /Axes/AxesTypeList.....	29
2006: /Axes/AxesType.....	30
2010: /Axes/MachineryPositions.....	31
2011: /Axes/AbsolutePositions.....	32
2012: /Axes/RelativePositions.....	33
2013: /Axes/DistanceToGo.....	34
2020: /Axes/LoadList.....	35
2021: /Axes/Load	36
3000: /Spindle/SpindleCount	37
3001: /Spindle/NameList.....	38
3002: /Spindle/Name.....	39
3010: /Spindle/ActualSpeedList.....	40
3011: /Spindle/ActualSpeed	41
3020: /Spindle/SpeedOverrideList	42
3021: /Spindle/SpeedOverride.....	43
3030: /Spindle/ActiveToolId.....	44
3031: /Spindle/ActiveToolOffsetValue	45



3040: /Spindle/LoadList.....	46
3041: /Spindle/Load.....	47
4000: /Plc/Fanuc	48
4010: /Plc/Siemens	49
4020: /Plc/Heidenhain	50
4030: /Plc/Mitsubishi	51
4040: /Plc/ItriGmc	52
4050: /Plc/Syntec	53
Support List.....	54



50: /Controller/Guid

編號	路徑	
50	/Controller/Guid	
參數型態		數值型態
Null		String
說明		
取得控制器 GUID。		
例：00000000-0000-0000-0000-XXXXXXXXXXXX		
C#範例：		
<pre>string guid; item.Path = "/Controller/Guid"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) guid = item.Value.ToString();</pre>		
使用條件		
無		
備註		
若機台支援 Ethernet 通訊，則末 12 碼即為網路卡 MAC 位置。		



60: /Controller/OperatingMode

編號	路徑
60	/Controller/OperatingMode
參數型態	數值型態
Null	Int
說明	
<p>取得控制器當前之加工模式。</p> <p>數值定義：</p> <ul style="list-style-type: none">● 0: JOG● 1: MDI● 2: MEM <p>C#範例：</p> <pre>int opMode; item.Path = "/Controller/OperatingMode"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) int.TryParse(item.Value.ToString(), out opMode);</pre>	
使用條件	
無	
備註	
無	



61: /Controller/OperatingStatus

編號	路徑
61	/Controller/OperatingStatus
參數型態	數值型態
Null	Int
說明	
<p>取得控制器當前之加工狀態。</p> <p>數值定義：</p> <ul style="list-style-type: none">● 0: Ready● 1: Feed hold● 2: Running● 3: Waiting● 4: Interrupted <p>C#範例：</p> <pre>int opStatus; item.Path = "/Controller/OperatingStatus"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) int.TryParse(item.Value.ToString(), out opStatus);</pre>	
使用條件	
無	
備註	
無	



62: /Controller/AlarmStatus

編號	路徑	
62	/Controller/AlarmStatus	
參數型態		數值型態
Null		Bool
說明		
<p>取得控制器當前之警報狀態。</p> <p>數值定義：</p> <ul style="list-style-type: none">● True: 警報發生● False: 無警報 <p>C#範例：</p> <pre>bool alarm; item.Path = "/Controller/AlarmStatus"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) bool.TryParse(item.Value.ToString(), out alarm);</pre>		
使用條件		
無		
備註		
無		



64: /Controller/RunTime

編號	路徑	
64	/Controller/RunTime	
參數型態		數值型態
Null		TimeSpan
說明		
取得控制器當前之累計開機時間。		
例：1.12:34:56		
C#範例：		
<pre>TimeSpan runTime; item.Path = "/Controller/RunTime"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) TimeSpan.TryParse(item.Value.ToString(), out runTime);</pre>		
使用條件		
無		
備註		
此資料會隨著控制器關機而重置。		



65: /Controller/OperationTime

編號	路徑
65	/Controller/OperationTime
參數型態	數值型態
Null	TimeSpan
說明	
取得控制器當前之累計加工時間。	
例：1.12:34:56	
C#範例：	
<pre>TimeSpan opTime; item.Path = "/Controller/OperationTime"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) TimeSpan.TryParse(item.Value.ToString(), out opTime);</pre>	
使用條件	
無	
備註	
此資料不會隨著控制器關機而重置。	



66: /Controller/CycleTime

編號	路徑	
66	/Controller/CycleTime	
參數型態		數值型態
Null		TimeSpan
說明		
取得控制器當前之加工時間。		
例：1.12:34:56		
C#範例：		
<pre>TimeSpan cycleTime; item.Path = "/Controller/CycleTime"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) TimeSpan.TryParse(item.Value.ToString(), out cycleTime);</pre>		
使用條件		
無		
備註		
此資料會隨著加工完成或中斷而重置，若當前非加工中狀態，則數值應固定為 00:00:00。		



67: /Controller/PartCount

編號	路徑
67	/Controller/PartCount
參數型態	數值型態
Null (回傳數值是完成工件數) Parameter[0]: Int (有參數時請參考說明)	Int
說明	
未提供參數值時，取得控制器當前之累計加工件數。例：123	
有提供參數值時， 當參數 Parameters[0]=0 時，取得控制器當前之累計加工件數。 當參數 Parameters[0]=1 時，取得控制器當前的需求工件數。	
C#範例 (未提供參數):	
<pre>int partCount; item.Path = "/Controller/PartCount"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) int.TryParse(item.Value.ToString(), out partCount);</pre>	
C#範例 (有提供參數):	
<pre>int partCount; item.Path = "/Controller/PartCount"; item.Parameter[0] = 1; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) int.TryParse(item.Value.ToString(), out partCount);</pre>	
使用條件	
無	
備註	
1. 此資料不會隨著控制器關機而重置，但控制器操作人員可手動重置。 2. 目前需求工件數僅支援 Fanuc。	



69: /Controller/RapidOverride

編號	路徑	
69	/Controller/RapidOverride	
參數型態		數值型態
Null		Double
說明		
取得控制器當前之快速進給速度百分比旋鈕數值。		
例：2000.0		
C#範例：		
<pre>double rapidOvr; item.Path = "/Controller/RapidOverride"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out rapidOvr);</pre>		
使用條件		
無		
備註		
數值單位為百分比(%)。		



70: /Controller/FeedRate

編號	路徑	
70	/Controller/FeedRate	
參數型態		數值型態
Null		Double
說明		
取得控制器當前之進給速度。		
例：300.0		
C#範例：		
<pre>double feedrate; item.Path = "/Controller/FeedRate"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out feedrate);</pre>		
使用條件		
無		
備註		
數值單位會隨著控制器設定不同而改變。		



71: /Controller/FeedRateOverride

編號	路徑
71	/Controller/FeedRateOverride
參數型態	數值型態
Null	Double
說明	
取得控制器當前之進給速度百分比旋鈕設定數值。	
例：85.0	
C#範例：	
<pre>double feedrateOvr; item.Path = "/Controller/FeedRateOverride"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out feedrateOvr);</pre>	
使用條件	
無	
備註	
數值單位為百分比(%)。	



72: /Controller/Program

編號	路徑	
72	/Controller/Program	
參數型態		數值型態
Null		String
說明		
取得控制器當前之加工程式名稱。		
例：00001		
C#範例：		
<pre>string program; item.Path = "/Controller/Program"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) program = item.Value.ToString();</pre>		
使用條件		
無		
備註		
無		



73: /Controller/Line

編號	路徑	
73	/Controller/Line	
參數型態		數值型態
Null		Int
說明		
取得控制器當前之加工行號。		
例：123		
C#範例：		
<pre>int line; item.Path = "/Controller/Line"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) int.TryParse(item.Value.ToString(), out line);</pre>		
使用條件		
無		
備註		
無		



80: /Controller/CurrentBlock

編號	路徑
80	/Controller/CurrentBlock
參數型態	數值型態
Null	String
說明	
取得控制器當前之單節命令。	
例：G01 X1000.0 F300.0	
C#範例：	
<pre>string currentBlock; item.Path = "/Controller/CurrentBlock"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) currentBlock = item.Value.ToString();</pre>	
使用條件	
無	
備註	
無	



81: /Controller/PreReadBlocks

編號	路徑
81	/Controller/PreReadBlocks
參數型態	數值型態
Null	String[]
說明	
取得控制器當前之預讀單節命令集，陣列中每一個元素皆代表未來的一個單節命令，依序排列。	
例：G01 X1000.0 F300.0	
C#範例：	
<pre>string[] prereadBlocks; item.Path = "/Controller/PreReadBlocks"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is string[]) prereadBlocks = item.Value as string[];</pre>	
使用條件	
無	
備註	
無	



82: /Controller/SpindleSpeedCmd

編號	路徑
82	/Controller/SpindelSpeedCmd
參數型態	數值型態
Null	Double
說明	
取得控制器當前之主軸轉速命令。	
例：S5000 => 5000	
C#範例：	
<pre>double speedCmd; item.Path = "/Controller/SpindleSpeedCmd"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out speedCmd);</pre>	
使用條件	
無	
備註	
數值單位會隨著控制器設定不同而改變。	



83: /Controller/FeedRateCmd

編號	路徑	
83	/Controller/FeedRateCmd	
參數型態		數值型態
Null		Double
說明		
取得控制器當前之進給速度命令。		
例：F300 => 300		
C#範例：		
<pre>double feedrateCmd; item.Path = "/Controller/FeedRateCmd"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out feedrateCmd);</pre>		
使用條件		
無		
備註		
數值單位會隨著控制器設定不同而改變。		



84: /Controller/ToolIdCmd

編號	路徑
84	/Controller/ToolIdCmd
參數型態	數值型態
Null	String
說明	
取得控制器當前之刀具命令。	
例：T3 => 3 例：T4DS2 => 4DS2	
C#範例：	
<pre>string toolCmd; item.Path = "/Controller/ToolIdCmd"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) toolCmd = item.Value.ToString();</pre>	
使用條件	
無	
備註	
無	



85: /Controller/ToolLengthOffsetCmd

編號	路徑
85	/Controller/ToolLengthOffsetCmd
參數型態	數值型態
Null	Double
說明	
取得控制器當前之刀具長度補正命令。	
例：H2 => 2	
C#範例：	
<pre>double offsetCmd; item.Path = "/Controller/ToolLengthOffsetCmd"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out offsetCmd);</pre>	
使用條件	
無	
備註	
無	



87: /Controller/GCodes

編號	路徑
87	/Controller/GCode
參數型態	數值型態
Null	String[]
說明	
取得控制器當前之 G Code 命令集，包含前墜 G 碼。	
例：G01 G28 G54	
C#範例：	
<pre>string[] gCode; item.Path = "/Controller/GCode"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is string[]) gCode = item.Value as string[];</pre>	
使用條件	
無	
備註	
無	



90: /Controller/Log/AlarmHistory

編號	路徑	
90	/Controller/Log/AlarmHistory	
參數型態		數值型態
Null		AlarmData[]
說明		
取得控制器當前警報記錄集，數值型態為 VMX 定義之 AlarmData 陣列，AlarmData 屬性如下：		
Properties name	Data type	Description
ID	String	Alarm number or ID
Message	String	Alarm message
AlarmTime	DateTime	Alarm time
C#範例：		
<pre>AlarmData[] alarmHistory; item.Path = "/Controller/AlarmHistory"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) { if (item.Value is AlarmData[]) { alarmHistory = item.Value as AlarmData[]; string almID = alarmHistory[0].ID; string almMsg = alarmHistory[0].Message; DateTime almTime = alarmHistory[0].AlarmTime; } }</pre>		
使用條件		
無		
備註		
依不同控制器廠牌型號，可能無法提供所有的資料項目。		



1800: /Compensation/ActiveWorkPieceOriginPointOffsetName

編號	路徑
1800	/Compensation/ActiveWorkPieceOriginPointOffsetName
參數型態	數值型態
Null	String
說明	
取得控制器當前之工作座標設定名稱。	
例：G54	
C#範例：	
<pre>string wpopName; item.Path = "/Compensation/ActiveWorkPieceOriginPointOffsetName"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) wpopName = item.Value.ToString();</pre>	
使用條件	
無	
備註	
無	



1801: /Compensation/WorkPieceOriginOffsetValue

編號	路徑
1801	/Compensation/WorkPieceOriginOffsetValue
參數型態	數值型態
Null	Double[]
說明	
取得控制器當前之工作座標設定數值。	
例：1.000 2.000 3.000 4.000 5.000	
C#範例：	
<pre>double[] wpooValue; item.Path = "/Compensation/WorkPieceOriginOffsetValue"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is double[]) wpooValue = item.Value as double[];</pre>	
使用條件	
無	
備註	
數值單位會隨著控制器設定不同而改變。	



1900: /Compensation/ToolOffsetFormat

編號	路徑
1900	/Compensation/ToolOffsetFormat
參數型態	數值型態
Null	String[]
說明	
取得路徑編號 1901 中，刀具補正資料表格之數值定義。	
例：Length Radius	
C#範例：	
<pre>string[] offsetFormat; item.Path = "/Compensation/ToolOffsetFormat"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is string[]) offsetFormat = item.Value as string[];</pre>	
使用條件	
無	
備註	
無	



1901: /Compensation/ToolOffsetTable

編號	路徑	
1901	/Compensation/ToolOffsetTable	
參數型態		數值型態
Null		ToolData[]
說明		
取得控制器當前之刀具資料表，數值型態為 VMX 定義之 ToolData 陣列，ToolData 屬性如下：		
Properties name	Data type	Description
ID	String	刀補編號
Values	Double[]	刀補數值，其順序與定義需由路徑編號 1900 取得。
C#範例：		
<pre>ToolData[] offsetTable; item.Path = "/Compensation/ToolOffsetTable"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is ToolData[]) { offsetTable = item.Value as ToolData[]; string toolId = offsetTable[0].ID; double[] toolOffsetValues = offsetTable[0].Values; }</pre>		
使用條件		
無		
備註		
數值單位會隨著控制器設定不同而改變。		



2000: /Axes/AxesCount

編號	路徑	
2000	/Axes/AxesCount	
參數型態		數值型態
Null		Int
說明		
取得機台進給軸數。		
例：5		
C#範例：		
<pre>int axesCount; item.Path = "/Axes/AxesCount"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) int.TryParse(item.Value.ToString(), out axesCount);</pre>		
使用條件		
無		
備註		
可能同時包含線性進給軸、旋轉進給軸、換刀機構、或其餘部件。		



2001: /Axes/AxesNameList

編號	路徑
2001	/Axes/NameList
參數型態	數值型態
Null	String[]
說明	
取得機台進給軸名列表。	
例：X Y Z B C	
C#範例：	
<pre>string[] axesNames; item.Path = "/Axes/NameList"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is string[]) axesNames = item.Value as string[];</pre>	
使用條件	
無	
備註	
進給軸順序會隨著機台設定不同而改變。	



2002: /Axes/AxesName

編號	路徑
2002	/Axes/Name
參數型態	數值型態
Parameter[0] : Int	String
說明	
根據進給軸索引編號取得機台之特定進給軸名。	
例：X	
C#範例：	
<pre>string axisName; item.Path = "/Axes/Name"; item.Parameter[0] = 1; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) axisName = item.Value.ToString();</pre>	
使用條件	
無	
備註	
進給軸順序與路徑編號 2001 取得之進給軸名列表資料相同，會隨著機台設定不同而改變。	



2005: /Axes/AxesTypeList

編號	路徑
2005	/Axes/TypeList
參數型態	數值型態
Null	String[]
說明	
取得機台進給軸類型列表。	
例：Linear Linear Linear Rotary Rotary	
C#範例：	
<pre>string[] axesTypes; item.Path = "/Axes/TypeList"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is string[]) axesTypes = item.Value as string[];</pre>	
使用條件	
無	
備註	
進給軸順序與路徑編號 2001 取得之進給軸名列表資料相同，會隨著機台設定不同而改變。	



2006: /Axes/AxesType

編號	路徑	
2006	/Axes/Type	
參數型態		數值型態
Parameter[0] : Int		String
說明		
根據進給軸索引編號取得機台之特定進給軸類型。		
例：Linear		
C#範例：		
<pre>string axisType; item.Path = "/Axes/Type"; item.Parameter[0] = 1; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) axisType = item.Value.ToString();</pre>		
使用條件		
無		
備註		
進給軸順序與路徑編號 2001 取得之進給軸名列表資料相同，會隨著機台設定不同而改變。		



2010: /Axes/MachineryPositions

編號	路徑
2010	/Axes/MachineryPositions
參數型態	數值型態
Null	Double[]
說明	
取得機台當前之進給軸機械座標。	
例：1.000 2.000 3.000 4.000 5.000	
C#範例：	
<pre>double[] machPos; item.Path = "/Axes/MachineryPositions"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is double[]) machPos = item.Value as double[];</pre>	
使用條件	
無	
備註	
進給軸順序與路徑編號 2001 取得之進給軸名列表資料相同，會隨著機台設定不同而改變。 數值單位會隨著控制器設定不同而改變。	



2011: /Axes/AbsolutePositions

編號	路徑
2011	/Axes/AbsolutePositions
參數型態	數值型態
Null	Double[]
說明	
取得機台當前之進給軸絕對座標。	
例：1.000 2.000 3.000 4.000 5.000	
C#範例：	
<pre>double[] absPos; item.Path = "/Axes/AbsolutePositions"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is double[]) absPos = item.Value as double[];</pre>	
使用條件	
無	
備註	
進給軸順序與路徑編號 2001 取得之進給軸名列表資料相同，會隨著機台設定不同而改變。 數值單位會隨著控制器設定不同而改變。	



2012: /Axes/RelativePositions

編號	路徑	
2012	/Axes/RelativePositions	
參數型態		數值型態
Null		Double[]
說明		
取得機台當前之進給軸相對座標。		
例：1.000 2.000 3.000 4.000 5.000		
C#範例：		
<pre>double[] relaPos; item.Path = "/Axes/RelativePositions"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is double[]) relaPos = item.Value as double[];</pre>		
使用條件		
無		
備註		
進給軸順序與路徑編號 2001 取得之進給軸名列表資料相同，會隨著機台設定不同而改變。 數值單位會隨著控制器設定不同而改變。		



2013: /Axes/DistanceToGo

編號	路徑
2013	/Axes/DistanceToGo
參數型態	數值型態
Null	Double[]
說明	
取得機台當前之進給軸餘位移量。	
例：1.000 2.000 3.000 4.000 5.000	
C#範例：	
<pre>double[] disToGo; item.Path = "/Axes/DistanceToGo"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is double[]) disToGo = item.Value as double[];</pre>	
使用條件	
無	
備註	
進給軸順序與路徑編號 2001 取得之進給軸名列表資料相同，會隨著機台設定不同而改變。 數值單位會隨著控制器設定不同而改變。	



2020: /Axes/LoadList

編號	路徑
2020	/Axes/LoadList
參數型態	數值型態
Null	Double[]
說明	
取得機台當前之進給軸負載百分比數值列表。	
例：10.0 20.0 30.0 40.0 50.0	
C#範例：	
<pre>double[] axesLoads; item.Path = "/Axes/LoadList"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is double[]) axesLoads = item.Value as double[];</pre>	
使用條件	
無	
備註	
進給軸順序與路徑編號 2001 取得之進給軸名列表資料相同，會隨著機台設定不同而改變。 數值單位為百分比(%)。	



2021: /Axes/Load

編號	路徑	
2021	/Axes/Load	
參數型態		數值型態
Parameter[0] : Int		Double
說明		
根據進給軸索引編號取得機台之特定進給軸負載百分比數值。		
例：10.0		
C#範例：		
<pre>double axisLoad; item.Path = "/Axes/Load"; item.Parameter[0] = 1; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out axisLoad);</pre>		
使用條件		
無		
備註		
進給軸順序與路徑編號 2001 取得之進給軸名列表資料相同，會隨著機台設定不同而改變。 數值單位為百分比(%)。		



3000: /Spindle/SpindleCount

編號	路徑	
3000	/Spindle/SpindleCount	
參數型態		數值型態
Null		Int
說明		
取得機台主軸數。		
例：2		
C#範例：		
<pre>int spindleCount; item.Path = "/Spindle/SpindleCount"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) int.TryParse(item.Value.ToString(), out spindleCount);</pre>		
使用條件		
無		
備註		
無		



3001: /Spindle/NameList

編號	路徑
3001	/Spindle/NameList
參數型態	數值型態
Null	String[]
說明	
取得機台主軸名列表。	
例：SP1 SP2	
C#範例：	
<pre>string[] spindleNames; item.Path = "/Spindle/NameList"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is string[]) spindleNames = item.Value as string[];</pre>	
使用條件	
無	
備註	
主軸順序會隨著機台設定不同而改變。	



3002: /Spindle/Name

編號	路徑
3002	/Spindle/Name
參數型態	數值型態
Parameter[0] : Int	String
說明	
根據主軸索引編號取得機台之特定主軸名。	
例：SP1	
C#範例：	
<pre>string spindleName; item.Path = "/Spindle/Name"; item.Parameter[0] = 1; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) spindleName = item.Value.ToString();</pre>	
使用條件	
無	
備註	
主軸順序與路徑編號 3001 取得之主軸名列表資料相同，會隨著機台設定不同而改變。	



3010: /Spindle/ActualSpeedList

編號	路徑
3010	/Spindle/ActualSpeedList
參數型態	數值型態
Null	Double[]
說明	
取得機台當前之主軸轉速列表。	
例：3000.0 5000.0	
C#範例：	
<pre>double[] speedList; item.Path = "/Spindle/ActualSpeedList"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is double[]) speedList = item.Value as double[];</pre>	
使用條件	
無	
備註	
主軸順序與路徑編號 3001 取得之主軸名列表資料相同，會隨著機台設定不同而改變。 數值單位會隨著控制器設定不同而改變。	



3011: /Spindle/ActualSpeed

編號	路徑	
3011	/Spindle/ActualSpeed	
參數型態		數值型態
Parameter[0] : Int		Double
說明		
根據主軸索引編號取得機台當前之特定主軸轉速。		
例：3000.0		
C#範例：		
<pre>double speed; item.Path = "/Spindle/ActualSpeed"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out speed);</pre>		
使用條件		
無		
備註		
主軸順序與路徑編號 3001 取得之主軸名列表資料相同，會隨著機台設定不同而改變。 數值單位會隨著控制器設定不同而改變。		



3020: /Spindle/SpeedOverrideList

編號	路徑
3020	/Spindle/SpeedOverrideList
參數型態	數值型態
Null	Double[]
說明	
取得機台當前之主軸轉速百分比旋鈕設定數值列表。	
例：90.0 100.0	
C#範例：	
<pre>double[] speedOvrs; item.Path = "/Spindle/SpeedOverrideList"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is double[]) speedOvrs = item.Value as double[];</pre>	
使用條件	
無	
備註	
主軸順序與路徑編號 3001 取得之主軸名列表資料相同，會隨著機台設定不同而改變。 數值單位為百分比(%)。	



3021: /Spindle/SpeedOverride

編號	路徑
3021	/Spindle/SpeedOverride
參數型態	數值型態
Parameter[0] : Int	Double
說明	
根據主軸索引編號取得機台當前之特定主軸轉速百分比旋鈕設定數值。	
例：90.0	
C#範例：	
<pre>double speedOvr; item.Path = "/Spindle/SpeedOverride"; item.Parameter[0] = 1; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out speedOvr);</pre>	
使用條件	
無	
備註	
主軸順序與路徑編號 3001 取得之主軸名列表資料相同，會隨著機台設定不同而改變。 數值單位為百分比(%)。	



3030: /Spindle/ActiveToolId

編號	路徑
3030	/Spindle/ActiveToolId
參數型態	數值型態
Null	String
說明	
取得機台當前之刀具編號或名稱。	
例：1 例：4DS2	
C#範例：	
<pre>string toolId; item.Path = "/Spindle/ActiveToolId"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) toolId = item.Value.ToString();</pre>	
使用條件	
1. *FANUC* 選刀的方式有順序選刀與任意選刀兩種。若是順序選刀，當前刀具編號在 CNC 畫面以 T 碼呈現，數值與刀具命令應相同；若是任意選刀，當前刀具編號在 CNC 畫面以 HDT 顯示，數值會存放在特定 PLC 內； <u>使用者必須在 CNC 連線設定中指定 HDTAddress 位址才能取得任意選刀模式下的刀號。</u>	
備註	
數值為機台刀庫中刀具的識別索引，根據控制器廠牌的不同，可能為編號或名稱。	



3031: /Spindle/ActiveToolOffsetValue

編號	路徑		
3031	/Spindle/ActiveToolIdOffsetValue		
參數型態		數值型態	
Null		Double	
說明			
取得控制器當前之刀具補正數值，數值型態為 VMX 定義之 ToolData，ToolData 屬性如下：			
Properties name		Data type	Description
ID		String	刀補編號
Values		Double[]	刀補數值，其順序與定義需由路徑編號 1900 取得。
C#範例：			
<pre>ToolData toolOffset; item.Path = "/Spindle/ActiveToolOffsetValue"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is ToolData) { toolOffset = (ToolData)item.Value; string toolId = toolOffset.ID; double[] toolOffsetValues = toolOffset.Values; }</pre>			
使用條件			
無			
備註			
數值單位會隨著控制器設定不同而改變。			



3040: /Spindle/LoadList

編號	路徑
3040	/Spindle/LoadList
參數型態	數值型態
Null	Double[]
說明	
取得機台當前之主軸負載百分比數值列表。	
例：15.0 20.0	
C#範例：	
<pre>double[] spindleLoads; item.Path = "/Spindle/LoadList"; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) if (item.Value is double[]) spindleLoads = item.Value as double[];</pre>	
使用條件	
無	
備註	
主軸順序與路徑編號 3001 取得之主軸名列表資料相同，會隨著機台設定不同而改變。 數值單位為百分比(%)。	



3041: /Spindle/Load

編號	路徑	
3041	/Spindle/Load	
參數型態		數值型態
Parameter[0] : Int		Double
說明		
根據主軸索引編號取得機台之特定主軸負載百分比數值。		
例：15.0		
C#範例：		
<pre>double spindleLoad; item.Path = "/Spindle/Load"; item.Parameter[0] = 1; int errorCode = cncAdaptor. ReadDataItem(ref item); if (errorCode == 0) double.TryParse(item.Value.ToString(), out spindleLoad);</pre>		
使用條件		
無		
備註		
主軸順序與路徑編號 3001 取得之主軸名列表資料相同，會隨著機台設定不同而改變。 數值單位為百分比(%)。		

**4000: /Plc/Fanuc**

編號	路徑
4000	/Plc/Fanuc
參數型態	數值型態
Parameter[0] : String Parameter[1] : String	根據參數的 PLC 型態。
說明	
存取 Fanuc 控制器之 PLC 數值。	
參數及數值的定義：	
參數索引	內容
PLC 型態	<p>根據 PLC 的設定可分別 4 種型態，格式是英文小寫字串。型態會決定讀取的數值型態。</p> <ul style="list-style-type: none"> • bit: 單一點，數值型態為 bool。設定為 ON 時，數值為 true。 • byte: 8 個 bits，數值為 byte。 • word: 16 個 bits，數值為 short。 • dword: 32 個 bits，數值為 int(C#)、long(C++)、或 Int32。
PLC 位址	<p>Fanuc PLC 位址，格式是 PLC 類型、位址與 Bit 位置。</p> <p>舉例如下：</p> <ul style="list-style-type: none"> • R2500.0 指的是 PLC 類型為 R，位址為 2500，Bit 位置是 0。欲讀取 bit 的資訊時，對應的 PLC 型態必須輸入為 bit。 • G12 指的是 PLC 類型為 G，位址為 12。若儲存資訊是以 byte 儲存時，對應的 PLC 型態必須輸入為 byte。
C#範例：	
讀取 PLC 的 bit 數值：	
<pre>item.Path = "/Plc/Fanuc"; item.Parameter[0] = "bit"; item.Parameter[1] = "R2500.1"; int errorCode = cncAdaptor.ReadDataItem(ref item); textbox.Text = item.Value.ToString(); //顯示該 PLC 點位之數值</pre>	
寫入 PLC 的 byte 數值：	
<pre>item.Path = "/Plc/Fanuc"; item.Parameter[0] = "byte"; item.Parameter[1] = "G12"; item.Value = 90; int errorCode = cncAdaptor.WriteDataItem(item);</pre>	
使用條件	
無	
備註	
細部設定請參考 Focas 說明文件。	



4010: /Plc/Siemens

編號	路徑
4010	/Plc/Siemens
參數型態	數值型態
Parameter[0] : String	請參考 Sinumerik 840D 通訊手冊
說明	
<p>存取 Siemens 控制器之 PLC 數值。</p> <p>參數定義： Sinumerik 定義之 PLC 路徑，例：DB99.DBX0.1。</p> <p>數值定義： 該 PLC 路徑所指向之數值，其型態根據路徑的不同可能為 Bool 或 Int。 詳細定義可參考 Sinumerik 840D 通訊手冊。</p> <p>C#範例（讀取）：</p> <pre>item.Path = "/Plc/Siemens"; item.Parameter[0] = "DB99.DBX0.1"; int errorCode = cncAdaptor.ReadDataItem(ref item); textbox.Text = item.Value.ToString(); //顯示該 PLC 點位之數值</pre> <p>C#範例（寫入）：</p> <pre>item.Path = "/Plc/Siemens"; item.Parameter[0] = "DB99.DBX0.1"; item.Value = 1; int errorCode = cncAdaptor.WriteDataItem(item);</pre>	
使用條件	
無	
備註	
無	



4020: /Plc/Heidenhain

編號	路徑	
4020	/Plc/Heidenhain	
參數型態		數值型態
Null		
說明		
使用條件		
無		
備註		
無		



4030: /Plc/Mitsubishi

編號	路徑	
4030	/Plc/Mitsubishi	
參數型態		數值型態
Parameter[0] : String		請參考 Mitsubishi 通訊手冊
說明		
<p>讀取 Mitsubishi 控制器之 PLC 數值。</p> <p>參數定義： Mitsubishi 定義之 PLC 路徑 例："YC10"，為 PLC 點位。</p> <p>數值定義： 呼叫 PLC 路徑所指向之點位資訊，其所使用之型別皆為 string。 詳細定義可參考 Mitsubishi CNC Communication Manual 通訊手冊。</p> <p>C#範例</p> <pre>String[] strUDevice = new String[ILength] strUDevice[0]="YC10"; 讀取該 PLC 點位資訊</pre>		
使用條件		
無		
備註		
無		

**4040: /Plc/ItriGmc**

編號	路徑
4040	/Plc/ItriGmc
參數型態	數值型態
Parameter[0] : String	int
說明	
<p>存取 GMC 控制器之 PLC 數值。</p> <p>參數定義： 定義讀取 PLC 之類型、起點以及數量[Type],[Start Index],[Bit Count]，三個數字以「，」分開。 Type: 0 -> I 點, 1 -> O 點, 2 -> C 點, 3 -> S 點, 4 -> A 點, 10 -> 強制 I 點, 11 -> 強制 O 點。 Start Index: 需要讀取 bit 的起點，第一個點起點指標為 0。 Bit Count: 欲讀取的 bit 數量，每個 PLC 點佔 1byte。</p> <p>例：0,100,1。(讀取 I 點，從 100，讀 1 PLC bit)</p> <p>數值定義： 每個 PLC 點佔 1byte。</p> <p>C#範例（讀取）：</p> <pre>item.Path = "/Plc/ItriGmc"; item.Parameter[0] = "0,100,1"; int errorCode = cncAdaptor.ReadDataItem(ref item); textbox.Text = item.Value.ToString(); //顯示該 PLC 點位之數值</pre> <p>寫入 GMC 控制器之 PLC 數值。</p> <p>參數定義： 定義寫入 PLC 之類型、起點以及數量[Type],[Start Index],[Bit Value]，三個數字以「，」分開。 Type: 3 -> S 點, 10 -> 強迫 I 點, 11 -> 強迫 O 點, 12 -> 強迫 C 點。 Start Index: 需要寫入 bit 的起點，第一個點起點指標為 0。 範圍限制：S 點 (有效數值為 200~299); 其它 (有效值為 0~實際最大點數-1)。 Bit Value: 欲寫入的值。 值限制：S 點 (0=OFF, 其他數值表示 ON); 其它 (0=該點不強制, 1=該點強制 ON, 2=該點強制 OFF)。</p> <p>例：11,100,1。(寫入 I 點，第 100 點，寫為 ON)</p>	



數值定義：

回傳值=0 或 回傳字串為「SUCCESS」代表成功，其它為錯誤。

C#範例（寫入）：

```
item.Path = "/Plc/ItriGmc";  
item.Parameter[0] = "11,100,1";  
int errorCode = cncAdaptor. WriteDataItem(item);
```

使用條件

無

備註

無

4050: /Plc/Syntec

編號	路徑	
4050	/Plc/Syntec	
參數型態		數值型態
Null		
說明		
使用條件		
無		
備註		
無		



Support List

- Fanuc: 30i, 18i, 0i, HSSB
- ITRI: GMC
- Siemens: 840D
- Mitsubishi: M70, M80, M700, M800
- Heidenhain: 530, 640
- Syntec: 21A-H

R: 上述型號皆支援讀取
W: 上述型號皆支援寫入
X: 不支援

ID	Fanuc	ITRI	Siemens	Mitsubishi	Heidenhain	Syntec	ID
50	X	R	R	X	X	x	50
60	R	R	R	R	R	R	60
61	R	R	R	R	X	R	61
62	R	R	X	X	X	x	62
64	R	R	R	R	X	x	64
65	R	R	R	R	X	x	65
66	R	R	R	R	X	x	66
67	R	R	R	R	X	x	67
70	R	R	R	R	R	R	70
71	R	R	R	R	R	R	71
72	R	R	R	R	R	R	72
73	X	R	R	R	R	x	73
80	R	R	R	R	R	x	80
81	R	R	R	R	R	x	81
82	R	R	R	R	R	R	82
83	R	R	R	R	R	R	83
84	R	R	R	R	R	R	84
85	R	R	R	R	R	R	85
87	R	R	X	X	X	x	87
90	R	X	X	X	X	x	90
1800	X	X	R	X	X	x	1800
1801	X	X	R	X	X	x	1801
1900	R	R	X	X	X	x	1900
1901	R	R	X	X	X	x	1901
2000	R	R	R	R	R	R	2000
2001	R	R	R	X	R	R	2001
2002	R	R	R	X	R	R	2002
2005	X	X	R	X	X	x	2005
2006	X	X	R	X	X	x	2006
2010	R	R	R	R	R	R	2010
2011	R	R	R	R	X	R	2011
2012	R	R	R	R	X	R	2012
2013	R	R	R	R	X	R	2013
2020	R	R	X	X	X	x	2020
2021	R	X	X	R	X	x	2021
3000	R	R	R	X	R	x	3000
3001	R	X	X	X	X	x	3001
3002	R	X	X	X	X	x	3002
3010	R	R	R	X	X	R	3010



ID	Fanuc	ITRI	Siemens	Mitsubishi	Heidenhain	Syntec	ID
3011	R	R	R	R	R	R	3011
3020	R	R	R	X	R	R	3020
3021	R	R	R	R	R	R	3021
3030	R	R	R	R	R	x	3030
3031	R	R	X	X	R	x	3031
3040	R	R	R	X	R	x	3040
3041	R	R	R	R	R	x	3041
4000	RW	x	x	x	x	x	4000
4010	x	R	x	x	x	x	4010
4020	x	x	RW	x	x	x	4020
4030	x	x	x	RW	x	x	4030
4040	x	x	x	x	RW	x	4040
4050	x	x	x	x	x	x	4050

Check version of API:

- Fanuc: 2.0.5.0
- ITRI: 1.0.0.0
- Siemens: 2.4.0.12
- Mitsubishi: 1.0.0.0
- Heidenhain: 1.1.0.0
- Syntec: 1.0.1.0