

# Prometheus - Getting Started

## 1 Installation

- Um eine Installation unter Windows NT, Windows 2000 oder Windows XP durchführen zu können, müssen Sie über Administratorrechte verfügen
- Legen Sie die CD ein und starten Sie *setup.exe* auf der CD
- Folgen Sie den Anweisungen
- Geben Sie als Seriennummer *eval-eval-eval* ein
- Vor dem ersten Start von Prometheus ist ein Neustart unbedingt erforderlich!

## 2 Deinstallation

Sie können Prometheus jederzeit wieder über die Systemsteuerung vollständig deinstallieren.

## 3 Zum ersten Projekt in 5 min.

Ein Beispiel soll demonstrieren, wie einfach es mit Prometheus ist, eine Anwendung zu erstellen, die eine Benutzeroberfläche und Zugriff auf Hardware bietet. Als "Hardware" dient hierbei der mitgelieferte Stecker für den Druckerport, der mit seinen beiden LEDs den Zustand zweier digitaler Ausgänge anzeigt.

### Aufgabe:

Es soll eine Applikation erstellt werden, die vom Benutzer eine Zeiteingabe verlangt und 2 LEDs am Druckerport blinken lässt.

### Lösung:

- Starten Sie Prometheus, ein Validationkey ist innerhalb der ersten 14 Tage nicht erforderlich
- Wählen Sie *Datei neu - Assistenten – Projektassistent*  
Es wurde der Projektassistent gestartet. Geben Sie für Projektname z.B. *LEDs* ein und wählen Sie den Pfad aus, unter dem die Projektdaten gespeichert werden sollen. Die übrigen Einstellungen bleiben. Drücken Sie auf *Fertigstellen*.
- Jetzt wurden alle notwendigen Dateien erzeugt:  
*LEDs.ppr* – Projekteinstellungen  
*LEDs.tpr* – Ablaufprogramm, Testprogramm  
*LEDs.phw* – Hardwarekonfiguration
- Als nächstes ist das Ablaufprogramm zu editieren, das Kern aller Prometheus-Applikationen ist. Das Programm, welches aus einfachen Anweisungen besteht, wird in Textform eingegeben. Bitte geben Sie folgende Zeilen in *LEDs.tpr* ein (die Kommentare – in kursiver Schrift – können auch weggelassen werden):

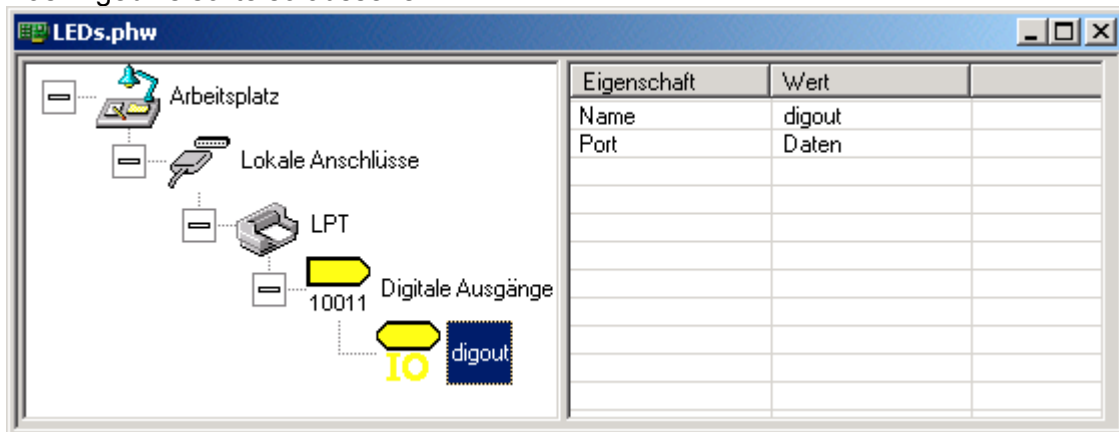
```
TimeVar Zeit;           //Variable deklarieren
Zeit = 500ms;          //Zeit vorbesetzen
Read("Dauer?", Zeit); //Benutzerabfrage
loop                   //Endlosschleife
{
SetDigital(digout, 1); //digitale Ausgänge schalten
ShowText("Rot");      //Text ausgeben
WaitTime(Zeit);       //Warten
SetDigital(digout, 2); //digitale Ausgänge schalten
ShowText("Grün");     //Text ausgeben
WaitTime(Zeit);       //Warten
}
```

- Nun fehlt nur noch die Definition und Konfiguration des digitalen Ausgangs *digout*. Dies ist im Fenster *LEDs.phw*, der Hardwarekonfiguration, vorzunehmen. Die Hardwarekonfiguration ist hierarchisch aufgebaut. Oberste Ebene ist Ihr *Arbeitsplatz*, im Moment beinhaltet er keine weiteren Elemente. Das Hinzufügen eines Elements wird folgendermaßen erledigt: Klicken Sie auf den Eintrag, dem etwas hinzugefügt werden soll, wählen Sie dann *Einfügen* aus dem *Menü Hardware* oder dem *Kontextmenü* (rechte Maustaste). Daraufhin erscheint ein Auswahldialog mit den verfügbaren Optionen. Option auswählen und mit OK bestätigen.

In unserem Beispiel:

Dem *Arbeitsplatz* die *lokalen Anschlüsse* hinzufügen,  
den *lokalen Anschlüssen* den *LPT* hinzufügen,  
dem *LPT* die *digitalen Ausgänge* hinzufügen.

Als letztes ist den *digitalen Ausgängen* ein *IOSymbol* zuzufügen und dessen Standardname von *IOSymbol* in *digout* (auf Eigenschaft *Name* klicken) zu ändern. Ein *IOSymbol* ist immer das unterste Glied in einer Hardwarekonfiguration. Das Ablaufprogramm greift ausschließlich über das *IOSymbol* auf die Hardware zu. Die Eigenschaft *Port* muss auf *Daten* gesetzt sein. Das Ergebnis sollte so aussehen:



- Das war es! Starten Sie die Applikation (*Kontextmenü*, *Menü Datei* oder klicken Sie auf den *grünen Pfeil*). Das Programm fragt den Benutzer zuerst nach einer Zeit. Nach der Bestätigung läuft das Programm endlos: Die LEDs leuchten im Wechsel und der aktuelle Blinkzustand wird im Ausgabefenster angezeigt.
- Die Applikation kann mit Stop beendet werden (in den Menüs oder Klick auf *rotes Quadrat*).

Dieses kleine Beispiel hat gezeigt, wie schnell und einfach ein Projekt erstellt, Zugriff auf Hardware gemanagt und ein Ablauf definiert werden kann. Die gleiche Vorgehensweise gilt auch für Zugriff auf komplexere Hardware, wie z.B. Messkarten, Netzgeräten oder Schnittstellen.

## Einige weitere Befehle

- für Hardwarezugriff
  - GetVoltage**(*IOSymbol*, *U*) – liest Spannung ein, z.B. von DMM oder PC-ADC-Karte
  - SendCAN**(*IOSymbol*, *ID*, *Daten*) – schickt ein CAN-Telegramm
  - GetCurrent**(*IOSymbol*, *I*) – liest einen Strom ein, z.B. von DMM oder Netzgerät
  - SetFrequency**(*IOSymbol*, *f*) – stellt eine Frequenz ein, z.B. am Funktionsgenerator
  - GetDigital**(*IOSymbol*, *byte*) – liest einen digitalen Eingang ein
- für Benutzeroberfläche
  - WaitBitmap**(*Datei*) – zeigt eine Grafik an und wartet auf Bestätigung
  - WaitText**(*Text*) – zeigt eine Meldung an und wartet auf Bestätigung
  - Say**(*Text*) – spricht einen Text über die Soundkarte
  - DoDialog**(*Name*) – führt einen vom Benutzer definierten Dialog aus
  - PlayMovie**(*Datei*) – spielt eine Videosequenz ab