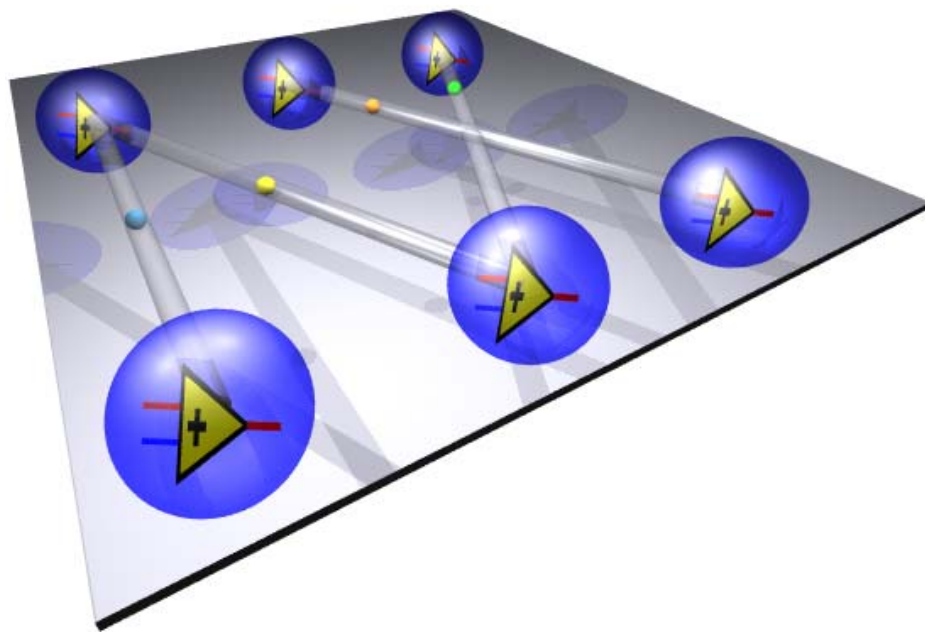


Tutorial

-

Class Inheritance Toolset



Erste Schritte in ObjectVIEW Professional

Anhand eines einfachen Beispiels zeigt dieses Tutorial schrittweise, wie man in ObjectVIEW eine neue Klasse erstellt und benutzt. Die Klasse erhält eigene Attribute, Icons und Methoden.

Es werden grundlegende Kenntnisse in LabVIEW und Objektorientierung vorausgesetzt.

Man stelle sich eine Schranke, z.B. am Eingang eines Betriebsgeländes vor.

Im Normalfall gibt sich der Fahrer eines ein- oder ausfahrenden Fahrzeuges zu erkennen, z.B. durch eine Chipkarte. Daraufhin öffnet sich die Schranke für kurze Zeit, so dass das Fahrzeug passieren kann. Dies soll im folgenden Automatikbetrieb heißen.

Wenn nötig, kann der Automatikbetrieb abgeschaltet werden. Dann bleibt die Schranke geöffnet, bis sie durch eine entsprechende Bedienhandlung geschlossen wird. Auch die Umschaltung in den Automatikbetrieb hat ein Schließen der Schranke zur Folge.

Die Schranke selbst kennt nur das Signal ‚geöffnet‘. Im aktiven Zustand (TRUE) veranlasst es die Schranke in die obere Endlage zu fahren und dort zu bleiben. Im entgegengesetzten Fall (geöffnet = FALSE) fährt die Schranke die untere Endlage an und verbleibt dort.

Von außen sollen folgende Zugriffe möglich sein:

- öffnen
- schließen
- Automatik ein/aus
- Auslesen des Status der Schranke

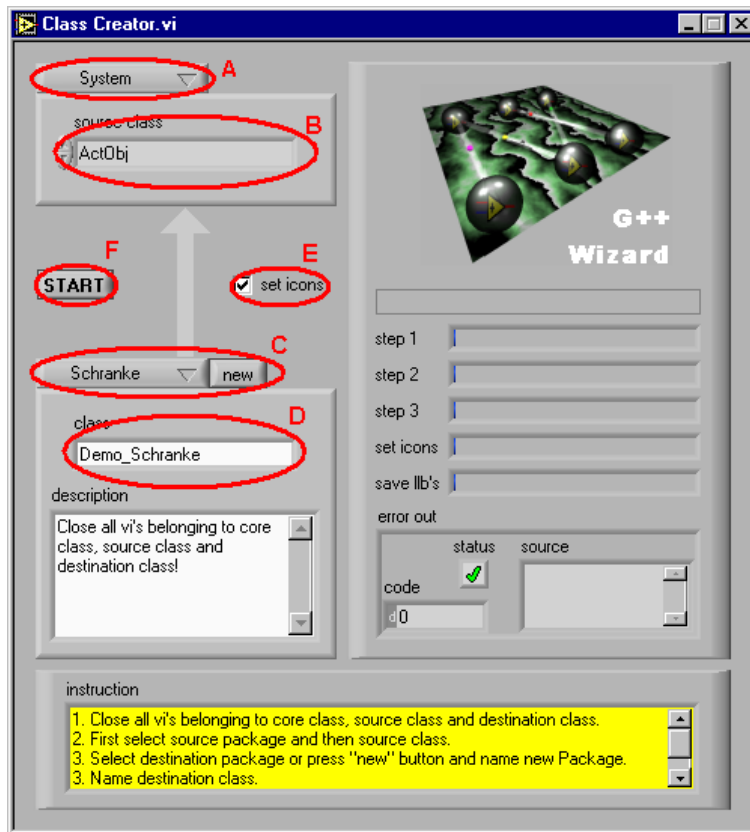
Die Klasse ‚Schranke‘ soll die oben beschriebene Funktionalität realisieren und ist mit den entsprechenden Methoden auszustatten. Der Status ist durch die Attribute

- *geöffnet*
- *auto*
- *Zeit*

definiert.

Ableiten einer Klasse

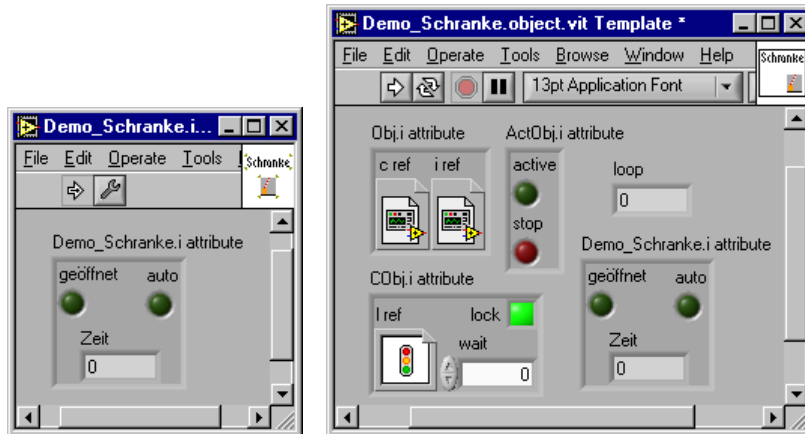
Der Class creator leitet neue Klassen von Systemklassen oder von Klassen, die Sie selbst erstellt haben, ab. Mit den folgenden Schritten erzeugen Sie die neue Klasse ‚Schranke‘.



- Öffnen Sie den Class Creator. (Tools -> Object – Technology -> Class Creator)
- Wählen Sie das Package der Oberklasse (System), von der die neue Klasse abgeleitet werden soll. (->A)
- Wählen Sie die Oberklasse ActObj. (->B)
- Die neue Klasse soll in einem neuen Package gespeichert werden. Klicken Sie den ‚new‘-Button an und geben Sie ‚Schranke‘ als Name des Packages an.(->C)
- Geben Sie den Namen ‚Demo_Schranke‘ für die neue Klasse ein. Sollten Sie mit einer Demoversion arbeiten, muss der Name mit ‚Demo_‘ beginnen. (->D)
- Um die Icons der VIs der neuen Klasse automatisch zu generieren, muss die Checkbox ‚set icons‘ (->E) markiert sein.
- Mit dem Start-Button (->F) beginnen Sie den Vererbungsprozess.

Nach einiger Zeit öffnen sich zwei Fenster: ‚Demo_Schranke.Object.vit‘ und ‚Demo_Schranke.i attribute.cnt‘

- Editieren Sie den Cluster ‚Demo_Schranke. i attribute‘ im Fenster ‚Demo_Schranke.i attribute.ctf‘. Er muss enthalten: ein Boolean ‚geöffnet‘, ein Boolean ‚auto‘ und einen Integer-Wert ‚Zeit‘.



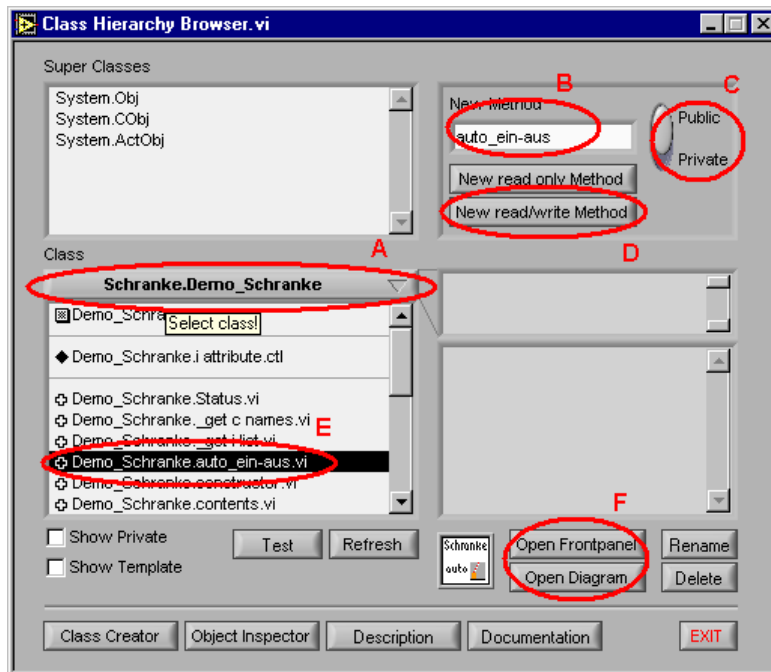
- Das Icon sollte ebenfalls angepasst werden. Der Inhalt der durch die grünen Winkel begrenzten Flächen wird in die Icons der neuen Klasse kopiert. Fügen Sie oben einen Schriftzug und unten ein einfaches Piktogramm ein.



- Die ‚Demo_Schranke.i attribute‘ müssen noch auf das Frontpanel des Objekt-Templates gelegt werden. Ziehen Sie das Icon des Fensters ‚Demo_Schranke.i attribute.cnt‘ auf das Frontpanel von ‚Demo_Schranke.Object.vit‘.
- Speichern und schließen sie ‚Demo_Schranke.i attribute.ctf‘ und ‚Demo_Schranke.Object.vit‘.
- Die Vererbung ist nach einiger Zeit beendet. Die Frage ‚Open Class Hierarchy Browser‘ beantworten Sie mit ‚Yes‘.

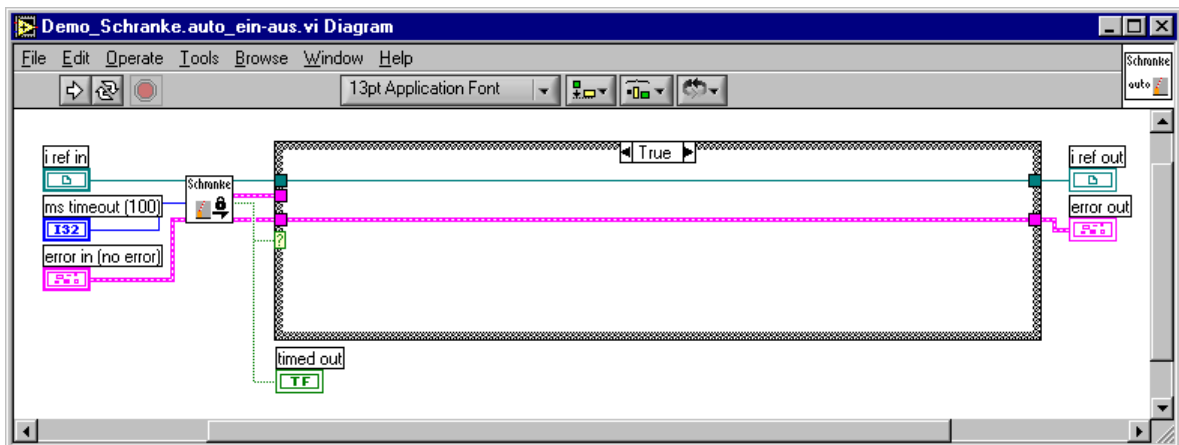
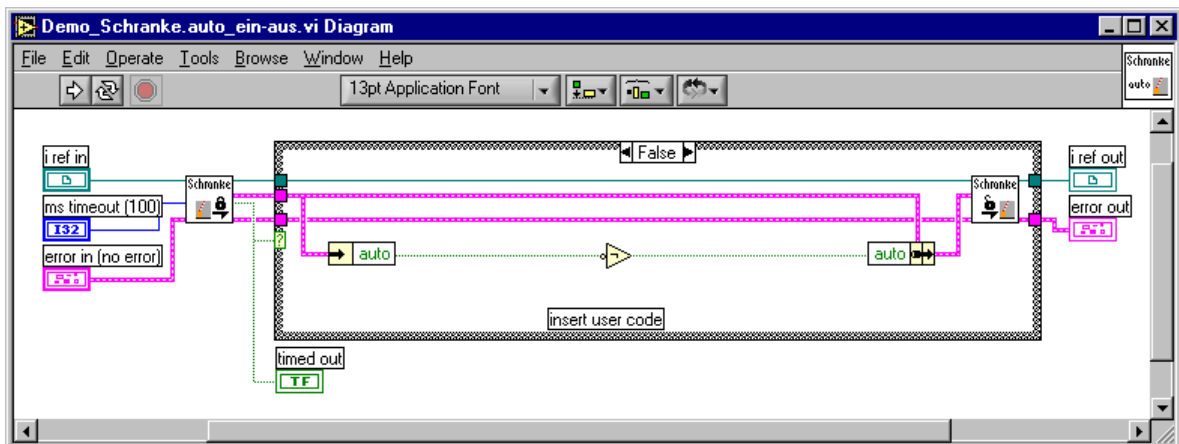
Neue Methoden erzeugen

Bei der Erzeugung neuer Methoden werden Sie durch den Class Hierarchy Browser unterstützt. Folgen Sie den Anweisungen, um die Klasse ‚Schranke‘ um die erforderlichen Methoden zu ergänzen.



- Wählen Sie die Klasse, die Sie bearbeiten möchten. (Schranke.Demo_Schranke, ->A)
Vor dem Punkt steht das Package, nach dem Punkt die Klasse.
- Tragen Sie ‚auto_ein-aus‘ als Name der neuen Methode in das Feld ‚New Method‘ ein (->B)
- Die Methode soll außerhalb des Objektes verwendet werden. Stellen Sie den Schalter deshalb auf ‚public‘(->C).
- Mit dem Button ‚New read/write Method‘ erzeugen Sie die Methode. Diese Methode greift unter wechselseitigem Auschluss auf ein Objekt der Klasse Schranke zu. In ihr können die Schranke.i attribute ausgewertet, manipuliert und zurückgeschrieben werden. (->D)

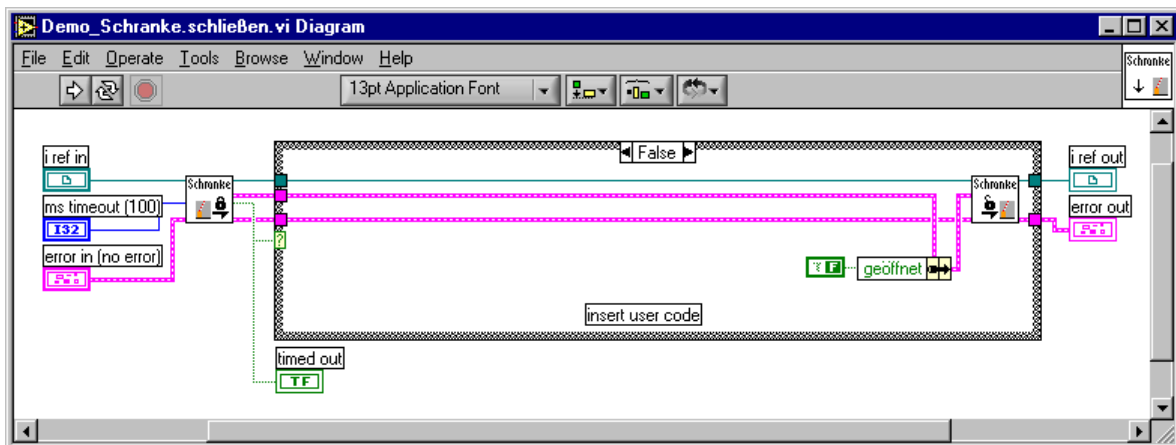
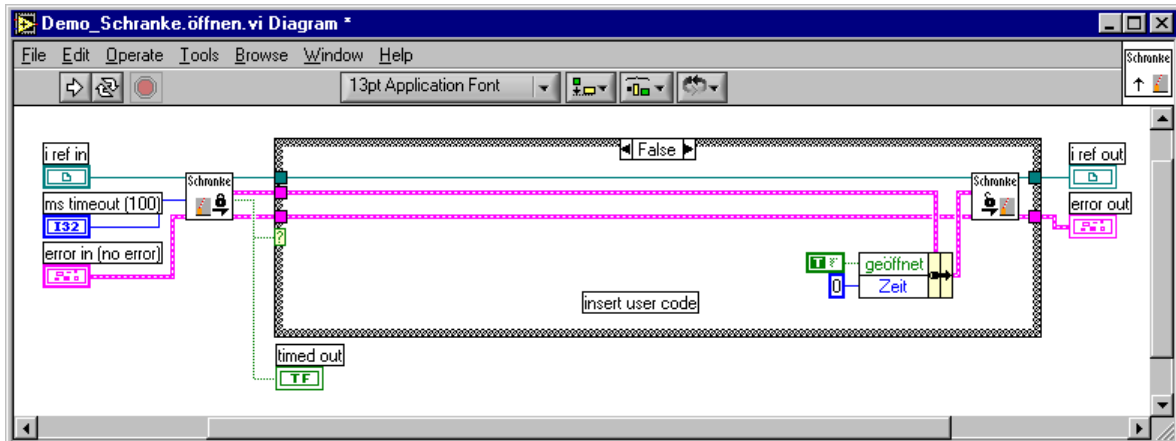
Auf dem Bildschirm erscheint das Frontpanel und das Blockdiagramm der Methode. Nehmen Sie die Änderungen vor, wie in der Abbildung ersichtlich.



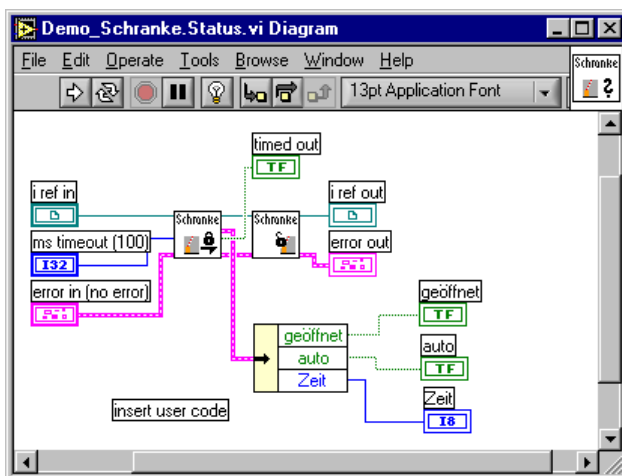
Bei einem Aufruf dieser Methode wird das Attribut *auto* gelesen, negiert und zurückgeschrieben, also der Automatikbetrieb ein oder ausgeschaltet. Währenddessen ist das Objekt für weitere Zugriffe gesperrt. Falls der Zugriff auf das Objekt nicht erfolgreich war, wird nichts getan (2. Abbildung)

- Mit dem Speichern und Schließen des Frontpanels der Methode übernehmen Sie die Änderungen.
- Wenn Sie weitere Änderungen an der Methode vornehmen wollen, wählen Sie sie in der VI-Liste aus (->E) und öffnen dann das Frontpanel/Blockdiagramm (->F)

Erzeugen Sie auf die gleiche Weise die Methoden ‚öffnen‘ und ‚schließen‘ mit den abgebildeten Diagrammen.

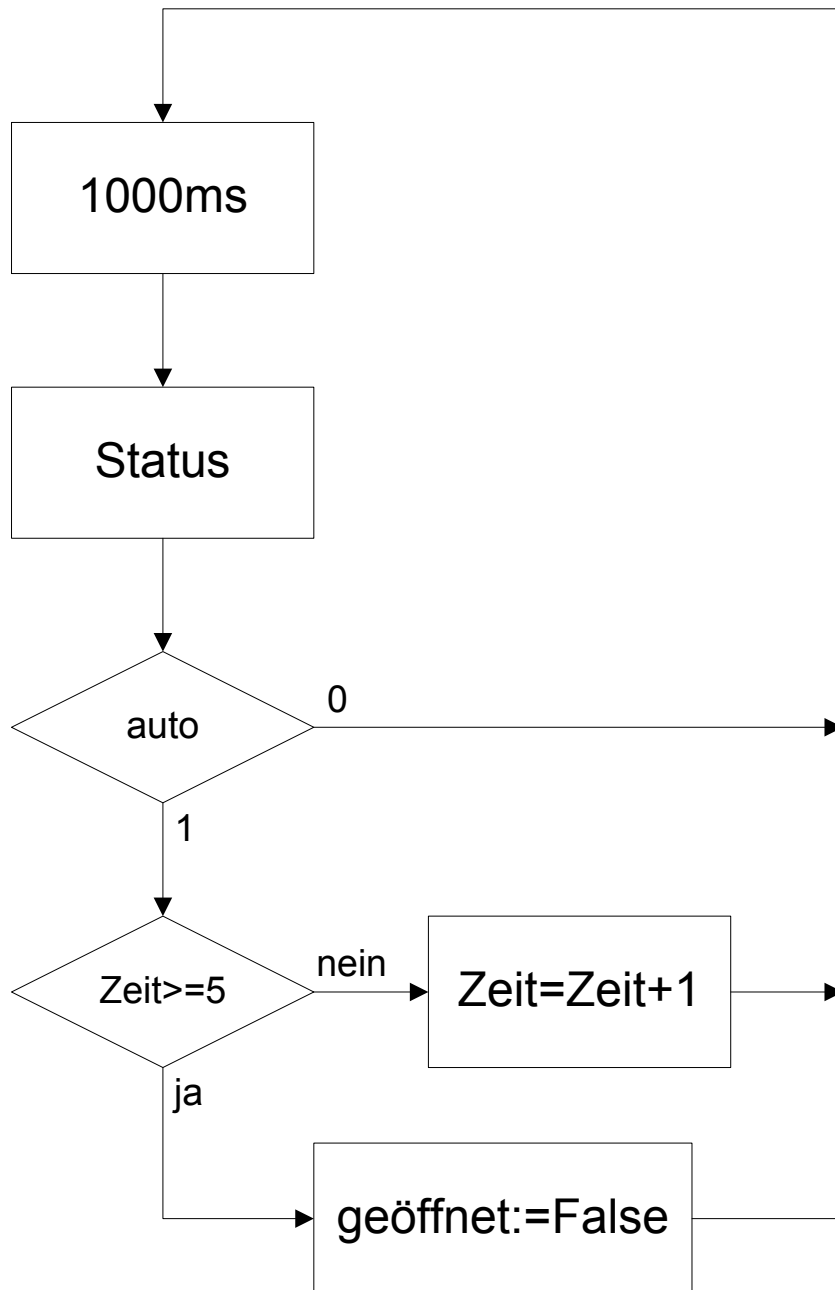


- Die Methode Status schreibt keine Daten zurück auf das Objekt. Sie ist mit dem Button ‚New read only method‘ zu erzeugen. Machen Sie die erforderlichen Änderungen im Diagramm (siehe Abb.). Die Indikatoren *geöffnet*, *auto* und *Zeit* sollen nach außen geführt werden und sind deshalb auf den Connector des VI zu legen.



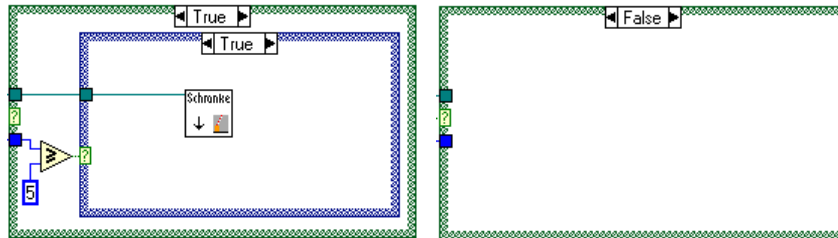
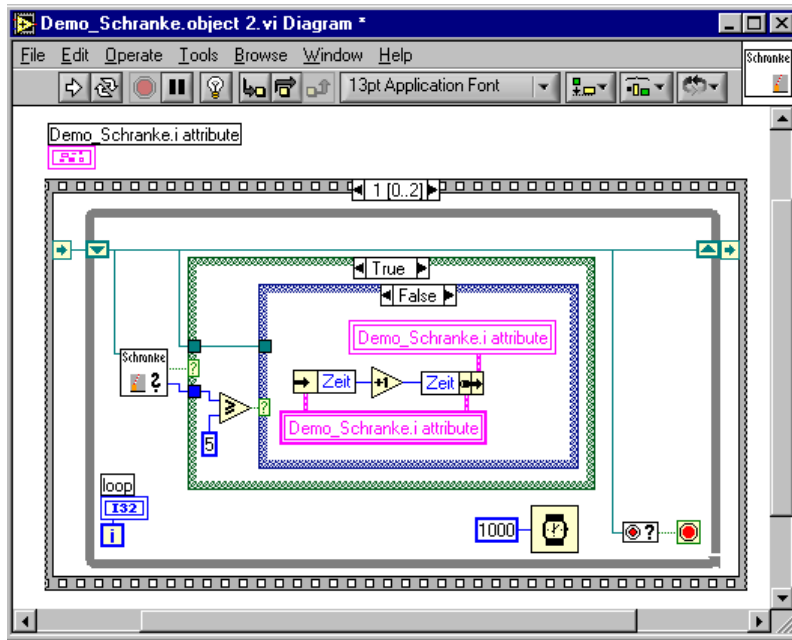
Erstellen des Objektprozesses

Die Verzögerung zwischen Öffnen und Schließen der Schranke im Automatikbetrieb wird durch einen objektinternen Prozess realisiert. Dieser liest periodisch alle 1000 Millisekunden den Status der Schranke und reagiert entsprechend. Dies ist in dem folgenden Diagramm dargestellt.



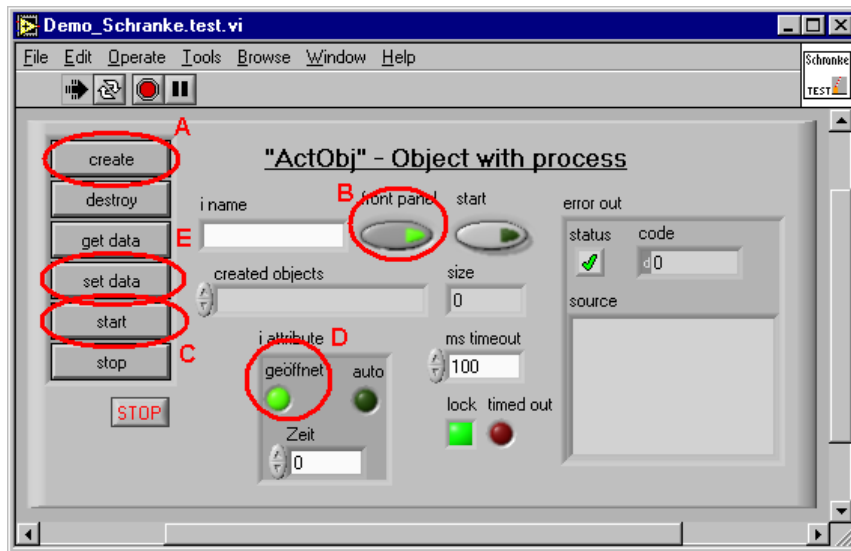
Objektinterne Prozesse sind nur in aktiven Objekten (ActObj, ScObj, EvtObj und EvtScObj sowie davon abgeleitete Klassen) möglich.

- Markieren Sie das Objekt-Template in der VI-Liste im Class Hierarchy Browser, öffnen Sie das Frontpanel und das Diagramm und führen Sie die in der Abb. dargestellten Veränderungen durch.



- Einer Methode muss die Referenz des aufgerufenen Objektes übergeben werden, um festzulegen, auf welches Objekt zugegriffen werden soll. Im Falle der Methoden ‚Status‘ und ‚schließen‘, ist das aufrufende Objekt selbst auch das Ziel des Aufrufes, also ist die eigene Referenz zu übergeben.
- Speichern und schließen Sie das Objekt-VIT.

Testen Sie das Objekt, indem Sie den Test-Button im Class Hierarchy Browser anklicken. Das erscheinende Programm kann Objekte erzeugen (create) und löschen (destroy), starten (start) und stoppen (stop) und Werte übergeben (set data) und auslesen (get data).



- Erzeugen und starten Sie ein Objekt. Das Frontpanel soll geöffnet sein. (->A, B,C)
- Im Cluster ‚i attribute‘ werden die zu übergebenden Werte eingegeben und mit ‚set data‘ übergeben. Aktivieren Sie ‚geöffnet‘ und klicken Sie auf ‚set data‘. (->D, E)
- Setzen Sie ‚geöffnet‘=FALSE, aktivieren Sie die Automatik (auto) usw. Funktioniert das Objekt wie erwartet?
- Vor beenden des Testprogramms sollte das Objekt entfernt werden (destroy).

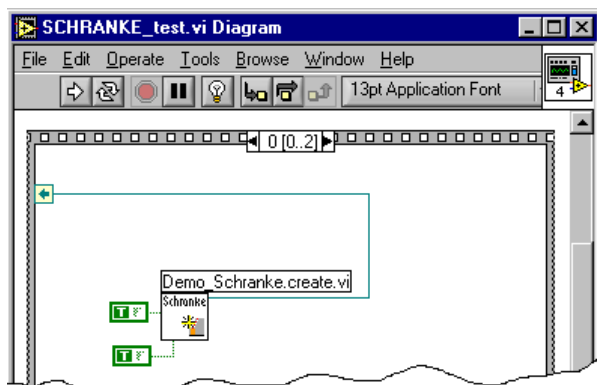
Test mit Speziellem Testprogramm

Mit dem Testprogramm, wie Sie es im vorhergehenden Abschnitt benutzt haben, kann nur die Funktion eines Objektes getestet werden, nicht jedoch die selbst programmierten Methoden. Es ist daher sinnvoll, ein kleines spezielles Testprogramm zu erstellen, das mit diesen Methoden auf ein Objekt der neuen Klasse zugreift.

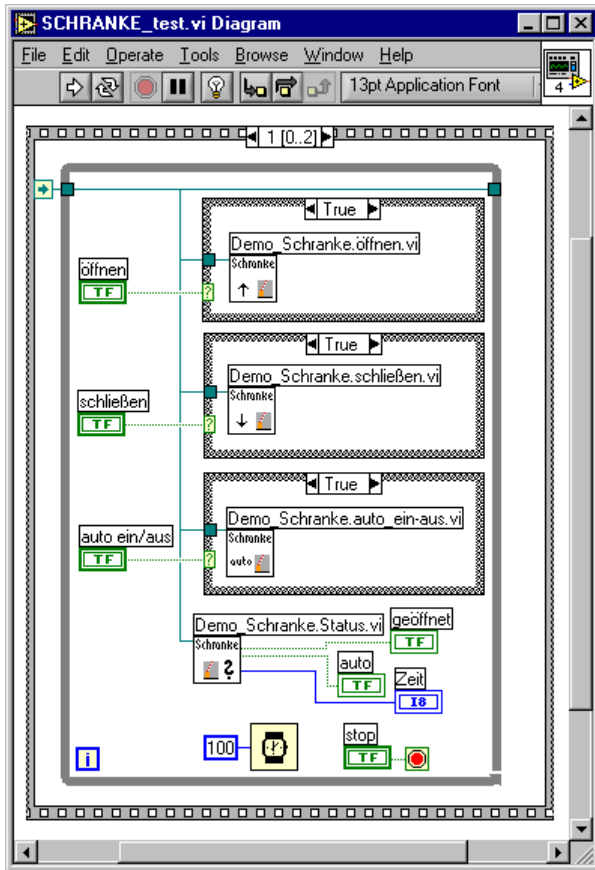
Das Testprogramm könnte z.B. wie in den folgenden Abbildungen aussehen.



Die Funktionsweise ist einfach. In Frame 0 erzeugt Demo_Schranke.create.vi ein Objekt mit geöffnetem Frontpanel. Die Referenz, des neuen Objektes wird dabei ausgegeben und auf ein ‚Sequence local‘ gelegt, da sie für die Methodenaufrufe benötigt wird.



Frame 1 enthält die Methodenaufrufe in einer Endlosschleife. Die Betätigung einer der Tasten hat die Abarbeitung des zugehörigen Case TRUE und damit des Methodenaufrufes zur Folge. ‚Schranke.Status.vi‘ wird immer abgearbeitet. Die Stop-Taste beendet die Schleife, so dass das Programm in den Frame 2 springt.



Frame 2 enthält das Löschen des Objektes mit Demo_Schranke.destroy.vi.

