Projektbeispiel Robotersteuerung

Grundlagen, Aufgabenstellung, Lösung

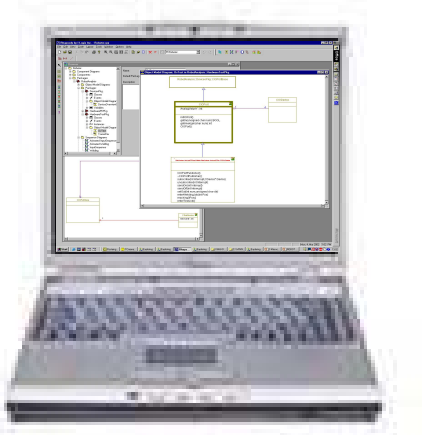
- **Wie wird ein UML Case-Tool in der Praxis eingesetzt ?**
- **Welche UML-Diagramme sind für die Codegenerierung notwendig ?**
- **Wie sieht die Unterstützung für Verifikation und Debugging aus ?**

Projektbeispiel:

Robotersteuerung

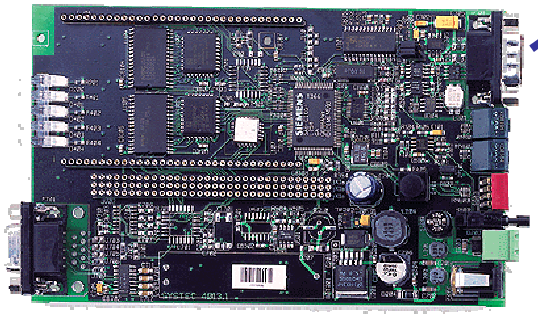
1. Grundlagen

Entwicklungsrechner mit UML Case-Tool und UML-Modell

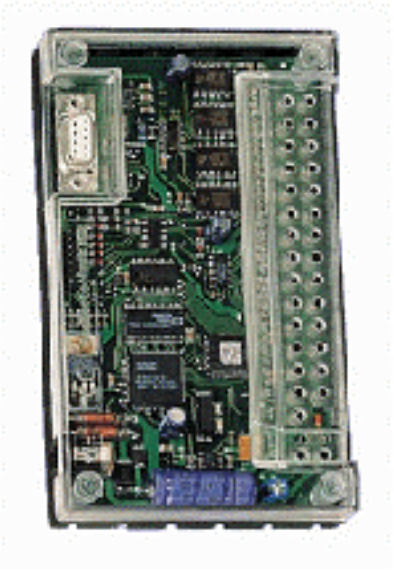


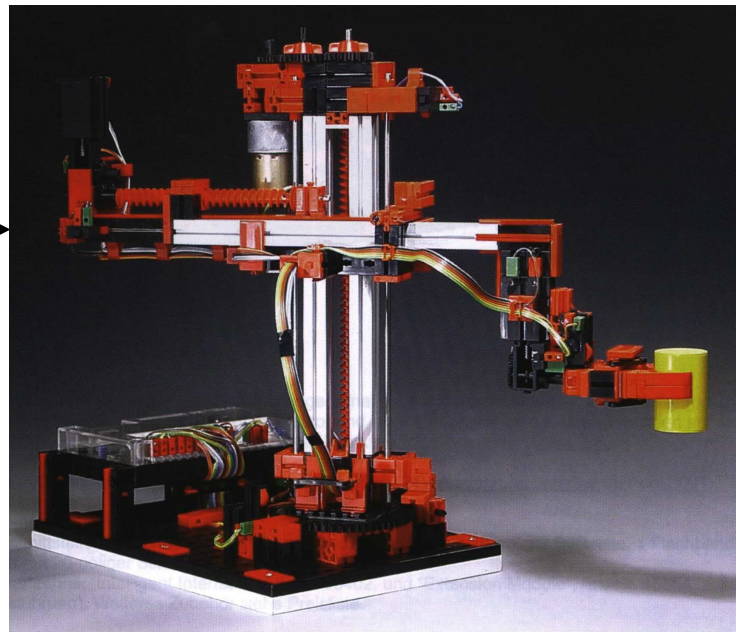
RS232
9600 Baud

TCP/IP
(optional)



optional

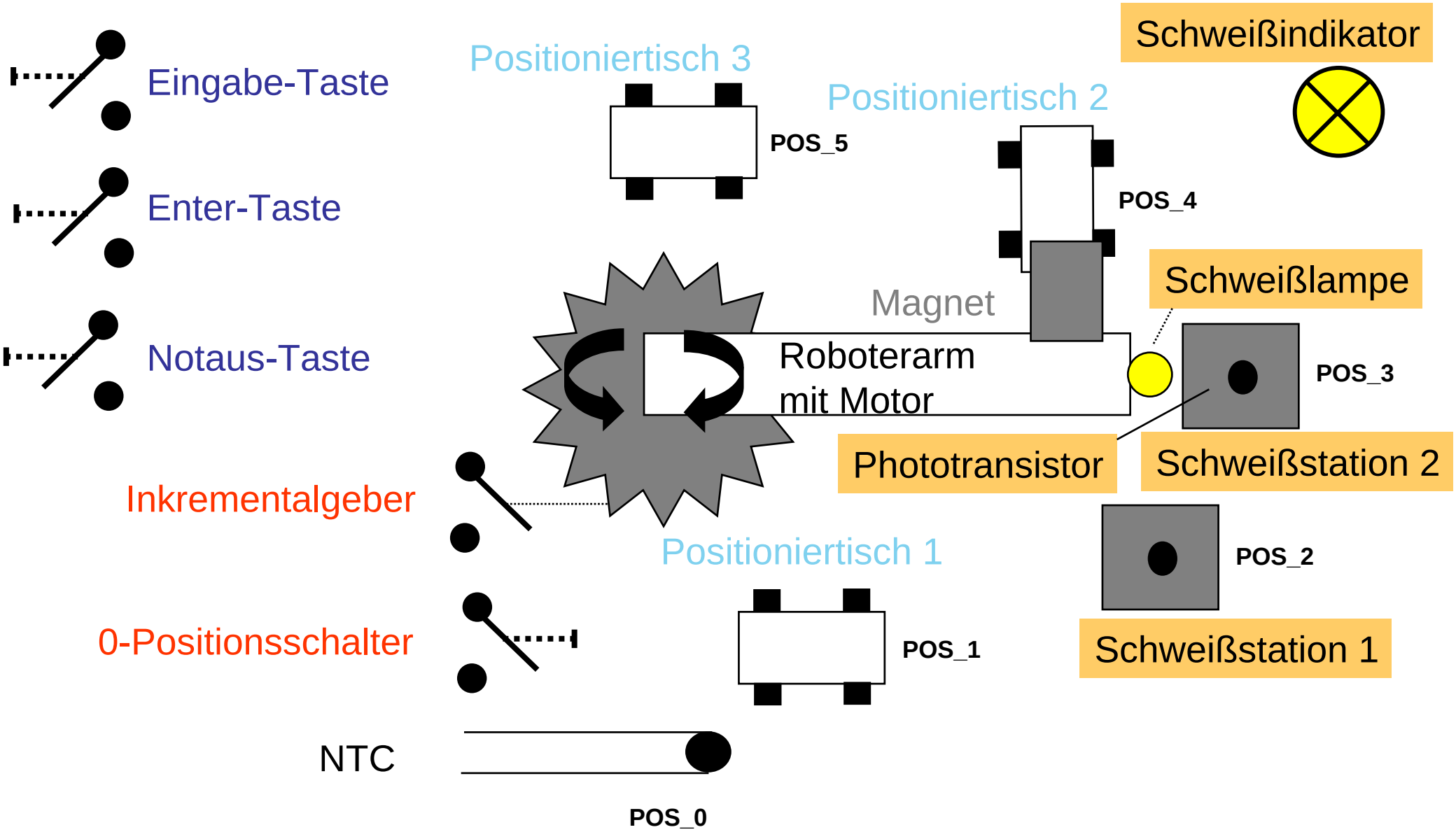


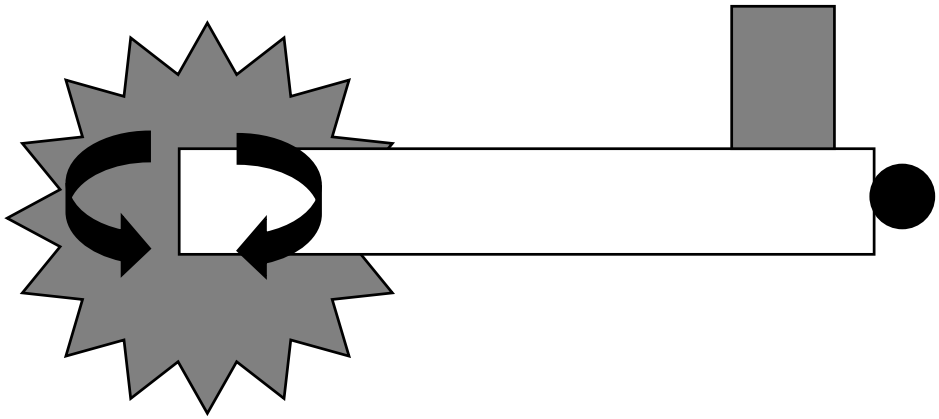


Embedded Hardware
z.B. C16x, ARM

Interface
z.B. Fischertechnik RS232<->TTL

Modell
z.B. Fischertechnik

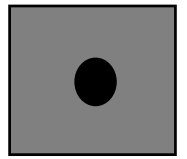




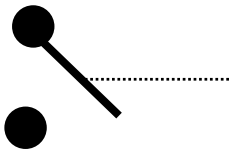
Motor: **M1**
Magnet: **M2**
Schweißindikator: **M3**
Schweißlampe: **M4**
NTC : **EY**

DeviceID = E[1..8]; M[1..4]
Interrupt = E[1..8] - 1

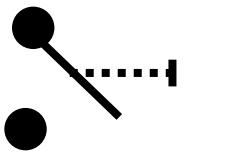
Schweißstation 2
Phototransistor: E3



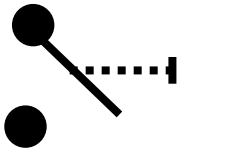
Inkrementalgeber: E2



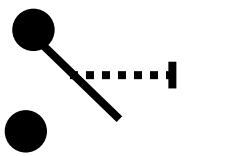
0-Positionsschalter: E1



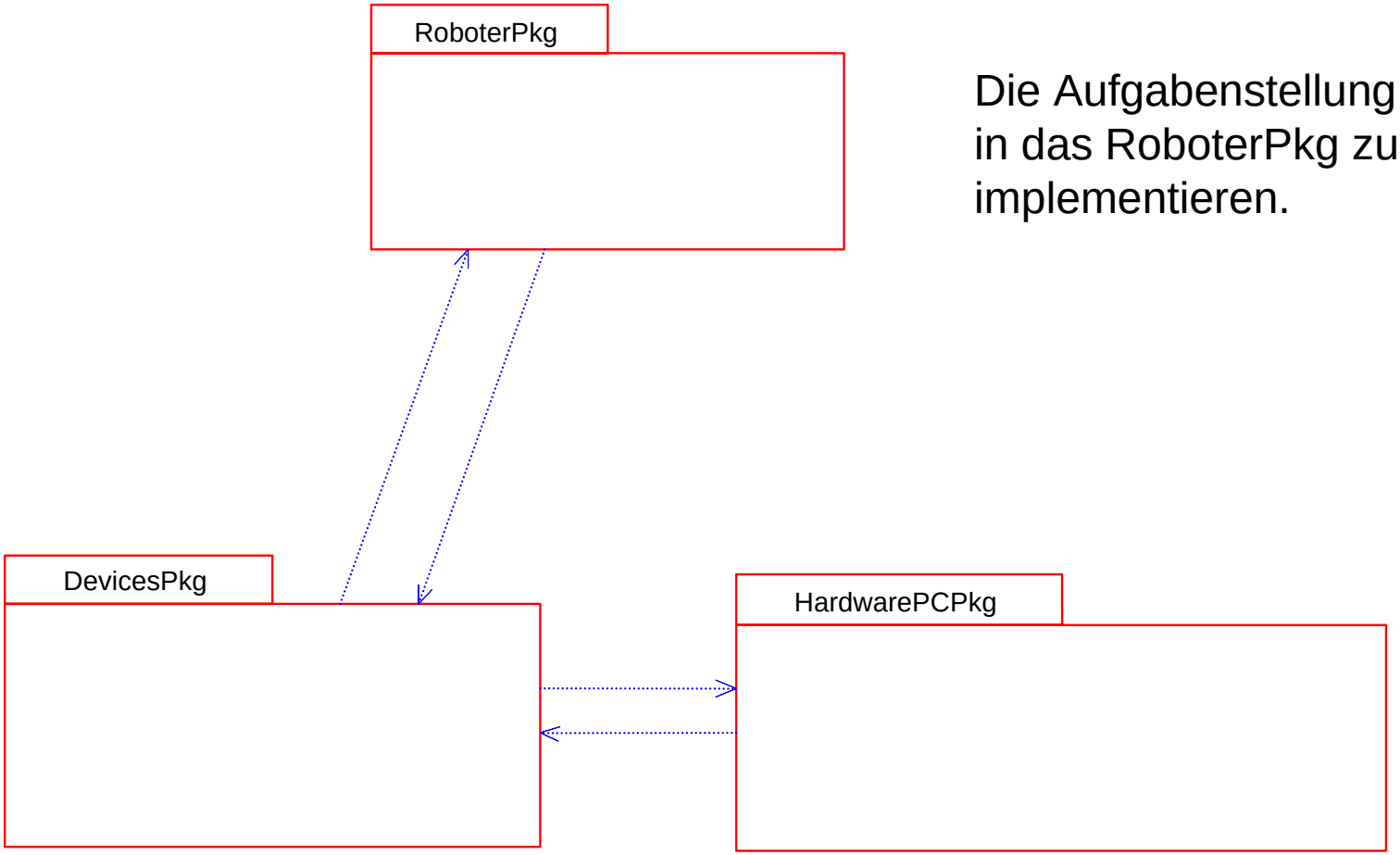
Enter-Taste: E6



Eingabe-Taste: E7

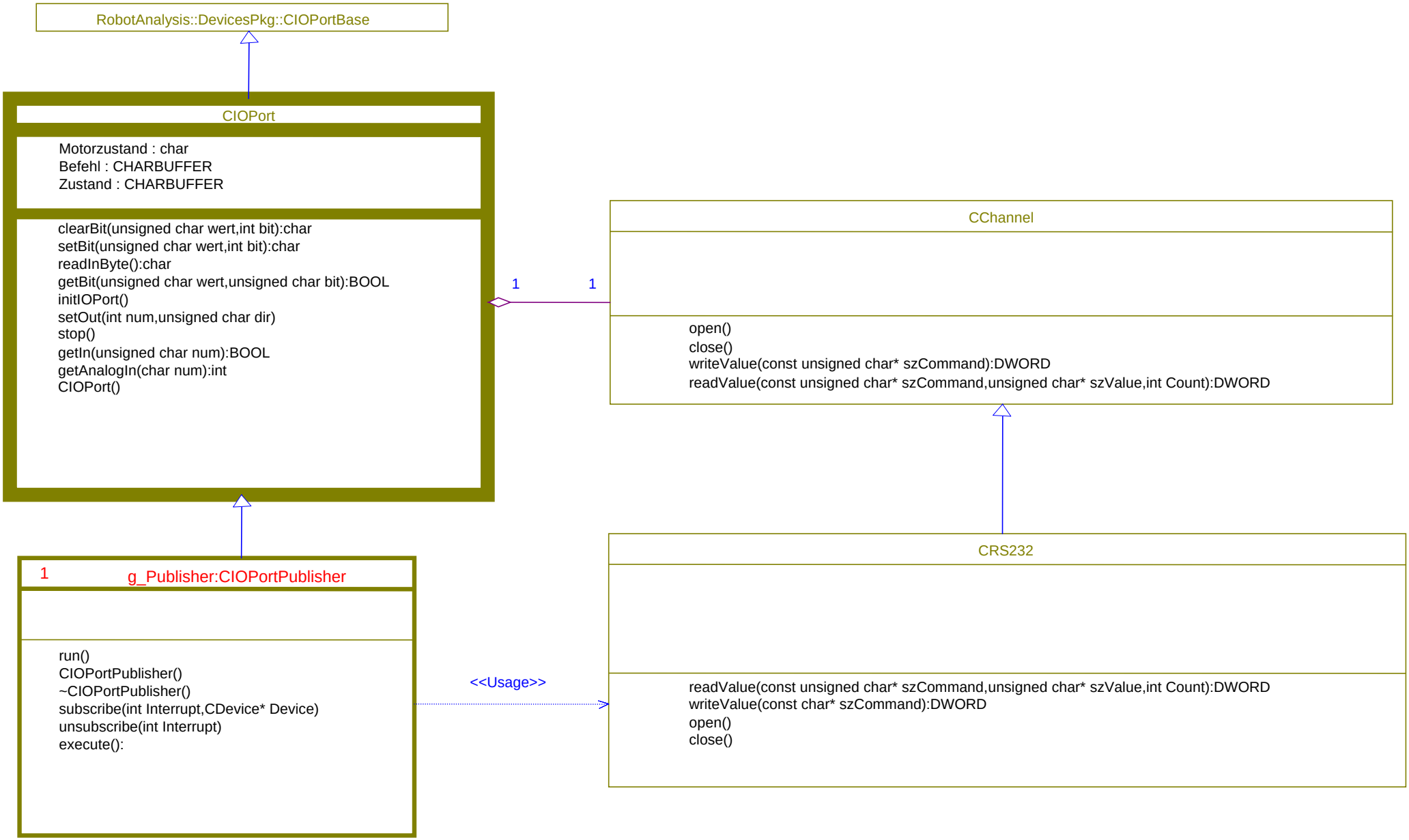


Notaus-Taste: E8

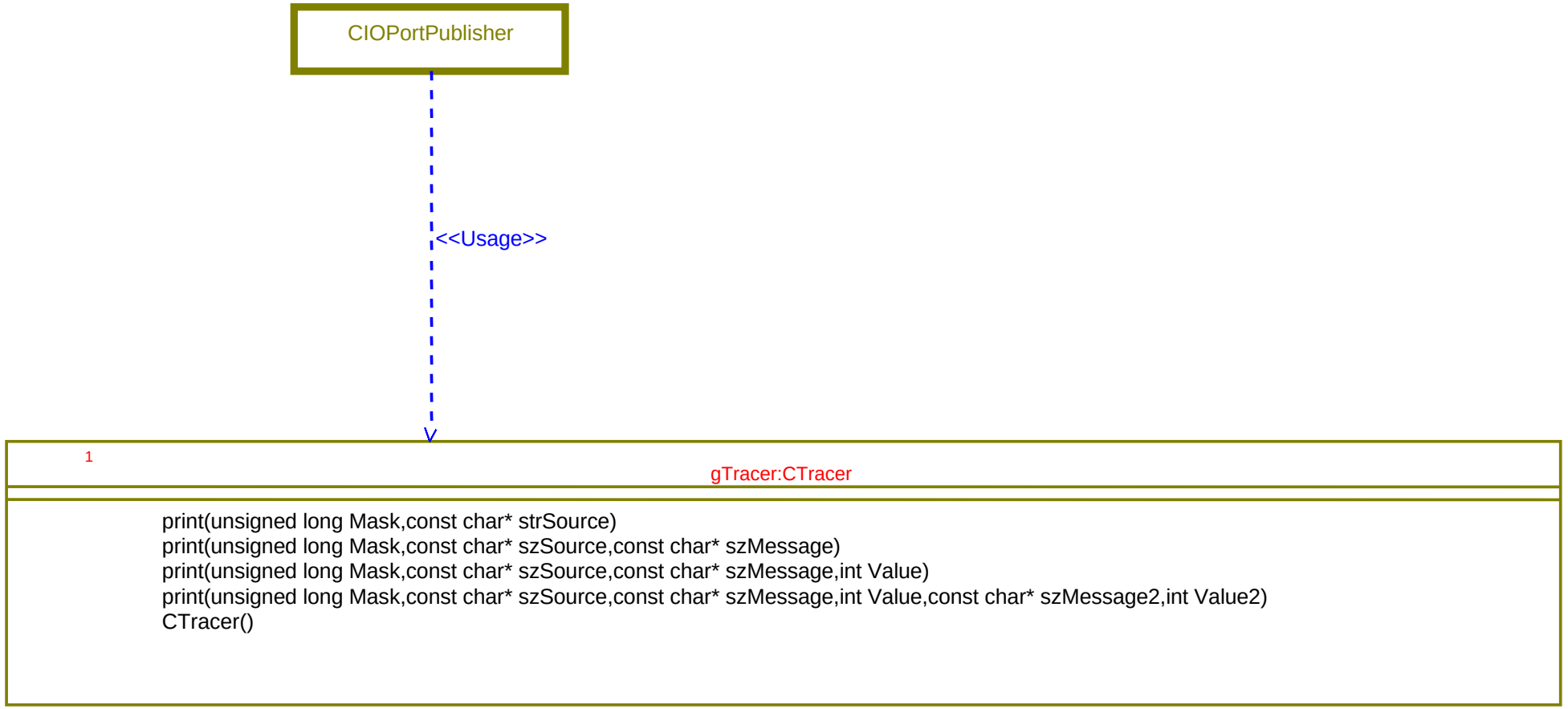


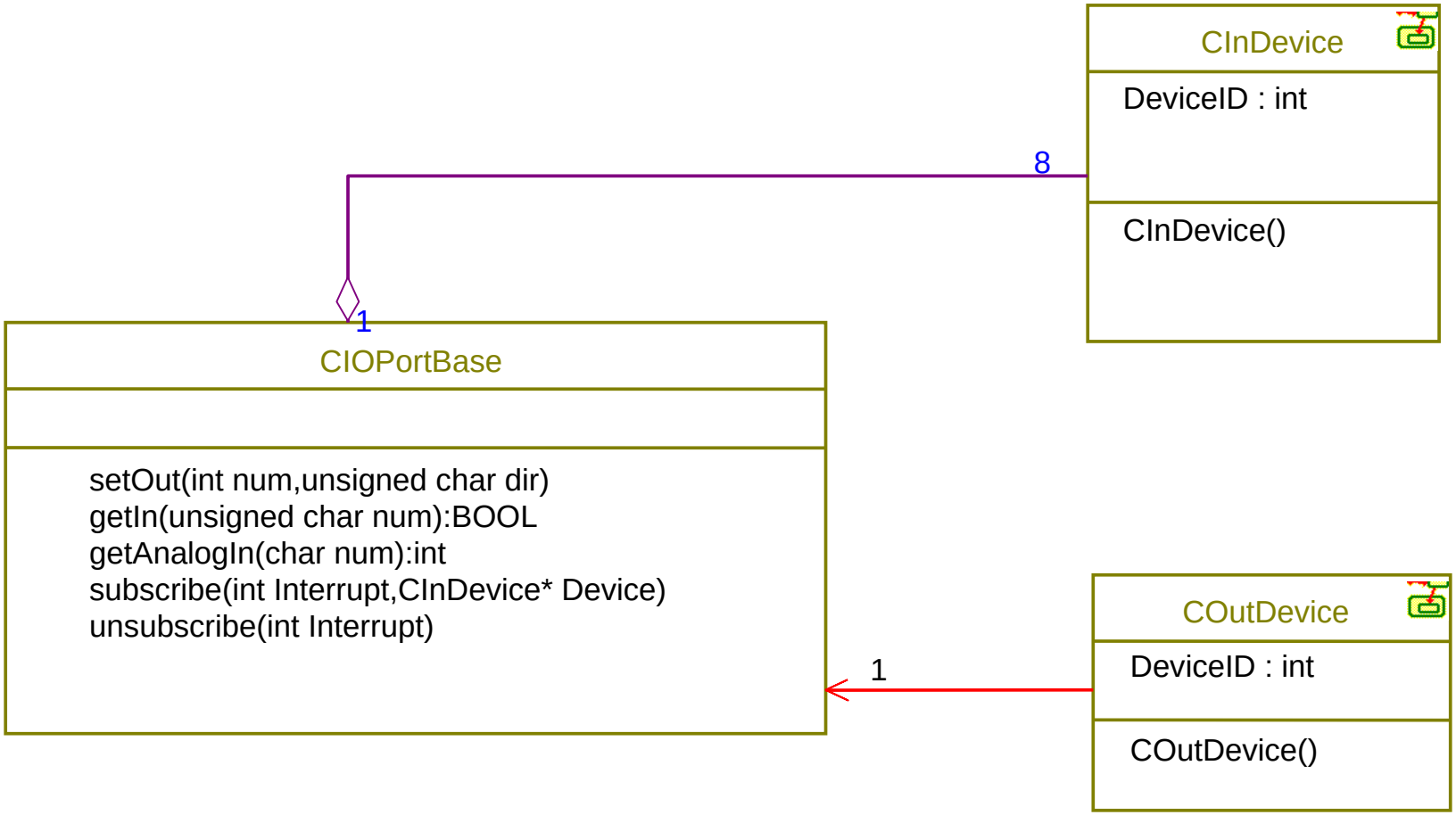
Die Aufgabenstellung ist in das RoboterPkg zu implementieren.

Robotersteuerung – Grundlagen: Paket HardwarePCPkg (1)



MICRO CONSULT





CIOPortBase-Interface für **Ausgaben**:

setOut(int num, unsigned char dir)

• int num	Ausgabeport M1..M4	[1; 2; 3; 4]
• dir	Drehrichtung	[´L´; ´R´; ´A´]

getIn(unsigned char num):BOOL

• unsigned char num	Eingabeport E1..E8	[1; 2; 3; 4; 5; 6; 7;8]

CIOPortBase-Interface für **Eingaben**:

getAnalogIn(char num):int

• char num	Analoger Eingabeport X; Y	[´X´; ´Y´]

subscribe(int Interrupt, CDevice* Device)

• int Interrupt	Interrupt für Eingabeport Interrupt = E[1..8] – 1	[0; 1; 2; 3; 4; 5; 6; 7]
• Cdevice* Device	Ereignissenke	[typisch this]

Generierte Ereignisse:

- evInterrupt_On
- evInterrupt_Off

Projektbeispiel:

Robotersteuerung

2. Aufgabenstellung

Durch Betätigung der Eingabe-Taste erfolgt eine kontinuierliche Rechtsdrehung des Roboterarms, solange die Taste gerückt bleibt. Wird die Taste wieder losgelassen, bleibt der Roboterarm an der entsprechenden Position stehen.

Durch Betätigung der Enter-Taste erfolgt eine kontinuierliche Linksdrehung des Roboterarms, solange die Taste gedrückt bleibt. Wird die Taste wieder losgelassen, bleibt der Roboterarm an der entsprechenden Position stehen.

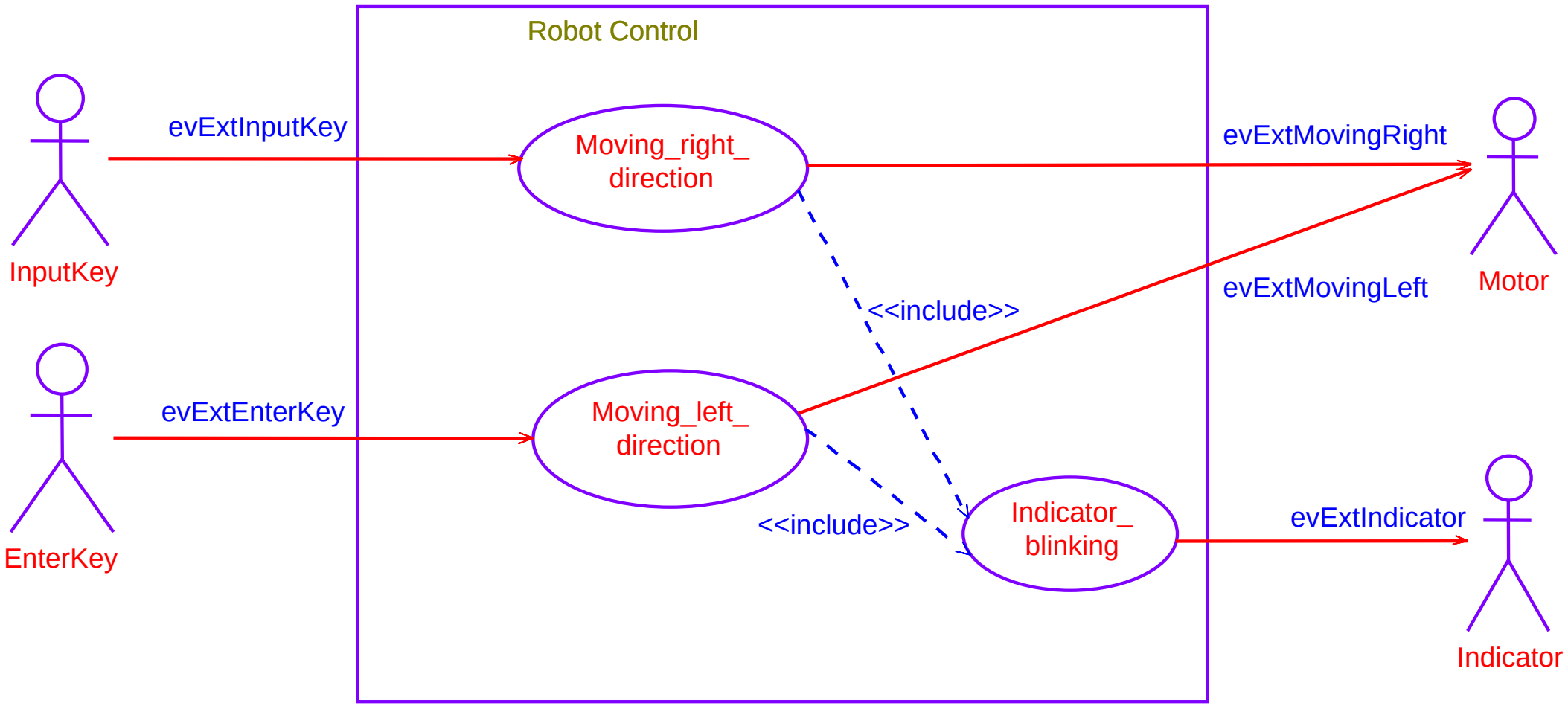
Während der Roboterarm in Bewegung ist, blinkt der Schweißindikator mit einem Takt von 500ms. Wenn der Roboterarm seine Drehbewegung beendet, muss der Schweißindikator aus sein.

Projektbeispiel:

Robotersteuerung

3. Lösung

Use-Case-Diagramm



Use-Case Spezifikation für Moving_right_direction

System:	**Robot Control**	
Akteur:	InputKey	
Use-Case:	Moving_right_direction	
Szenarien:	Input key is pressed	
Vorbedingungen:	System is running error-free	
Ablaufsequenz:		**Exception:**
The user presses the input key		-
The motor starts running in the right direction until the user releases the input key.		-
Nachbedingungen:	-	
Referenzen:	Use-Case-Diagram „Overview“	

Use-Case Spezifikation für Moving_left_direction

System:	**Robot Control**	
Akteur:	EnterKey	
Use-Case:	Moving_left_direction	
Szenarien:	Enter key is pressed	
Vorbedingungen:	system is running error-free	
Ablaufsequenz:		**Exception:**
The user presses the enter key		-
The motor starts running in the left direction until the user releases the enter key.		-
Nachbedingungen:	-	
Referenzen:	Use-Case-Diagram „Overview“	

Use-Case Spezifikation für Indicator_blinking

System:	**Robot Control**	
Akteur:	Indicator	
Use-Case:	Indicator_blinking	
Szenarien:	The robot rotates in the right or left direction	
Vorbedingungen:	System is running error-free	
Ablaufsequenz:		**Exception:**
The indicator is blinking automatically.		If the Motor stops running, then the indicator also stops blinking.
Nachbedingungen:	-	
Referenzen:	Use-Case-Diagram „Overview“	

Beschreibung der systemexternen Ereignisse

	Ereignis	Beschreibung	Richtung	Art	Antwortzeit
1	**evExtInputKey**	Löst eine Rechtsbewegung des Roboterarms aus	Initiator ist der Akteur InputKey	Asynchron, nicht periodisch	unkritisch
2	**evExtEnterKey**	Löst eine Linksbewegung des Roboterarms aus	Initiator ist der Akteur EnterKey	Asynchron, nicht periodisch	unkritisch
3	**evExtMovingRight** **evExtMovingLeft**	Ansteuerung des Motors für die Links-/Rechts-bewegung	Initiator ist der Use-Case Moving_left/right_direction	Asynchron, konstantes Signal	unkritisch
4	**evExtIndicator**	Steuert den Indikator an	Initiator ist der Use-Case Indicator_blinking	synchron, periodisch	500ms

Von den Objekten in der Realität zu den Klassen im Softwaremodell

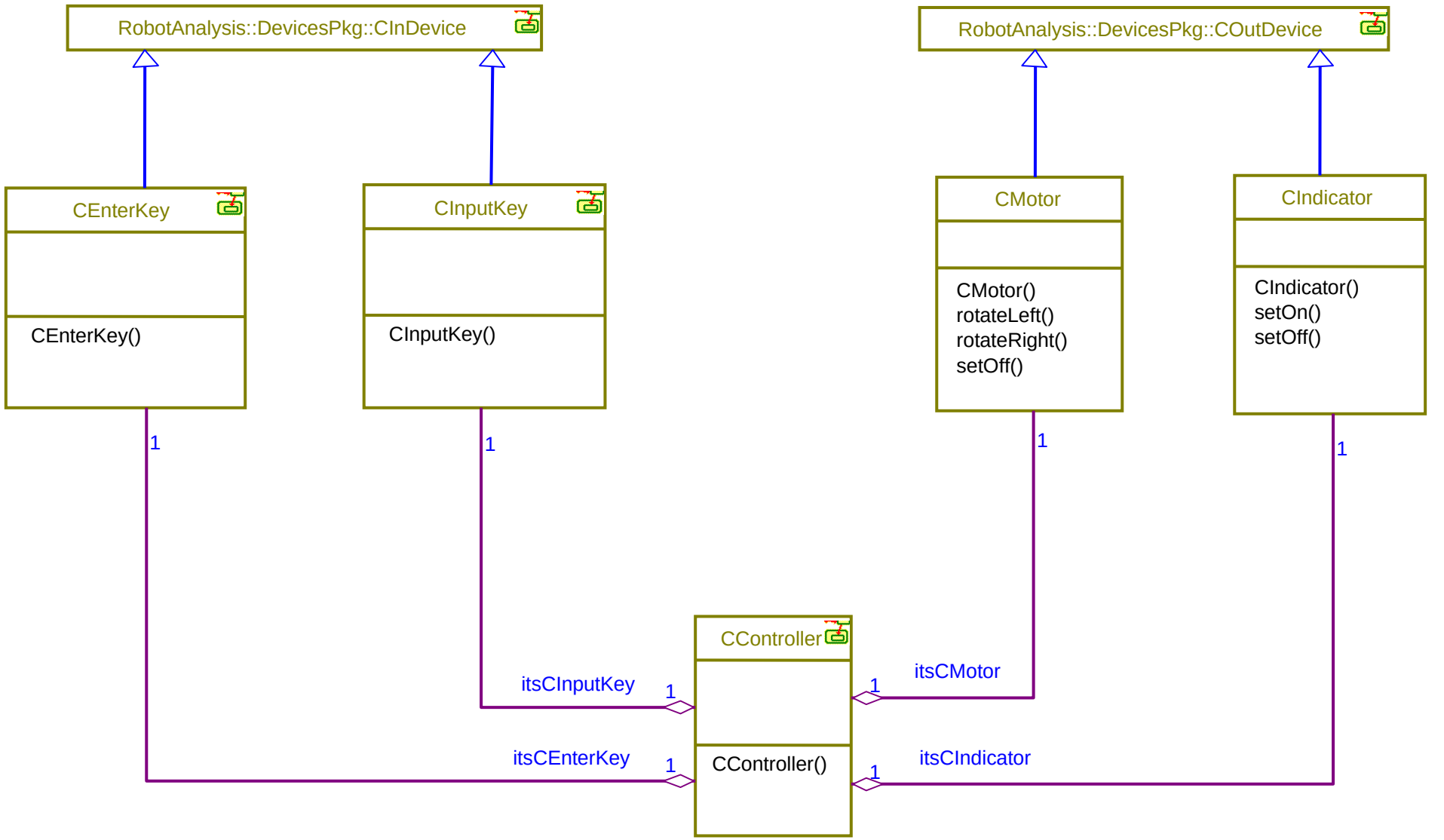
Objekte in der Realität:
- Eingabe-Taste
- Enter-Taste
- Indikator
- Motor

Klassen im Softwaremodell:
- CInputKey
- CEnterKey
- CIndicator
- CMotor
- CController

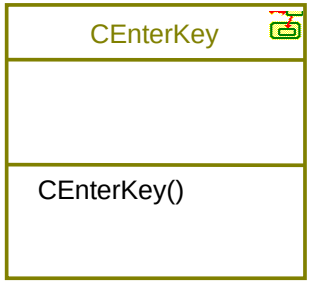
Hinweis:
In diesem einfachen Beispiel werden alle Objekte 1:1 von der Realität in dem Softwaremodell als Klasse abgebildet. Dies ist aber nicht immer der Fall. Auch hier wäre es möglich, die Objekte Eingabe-Taste und Enter-Taste gemeinsam in einer Klasse CKey zu beschreiben.

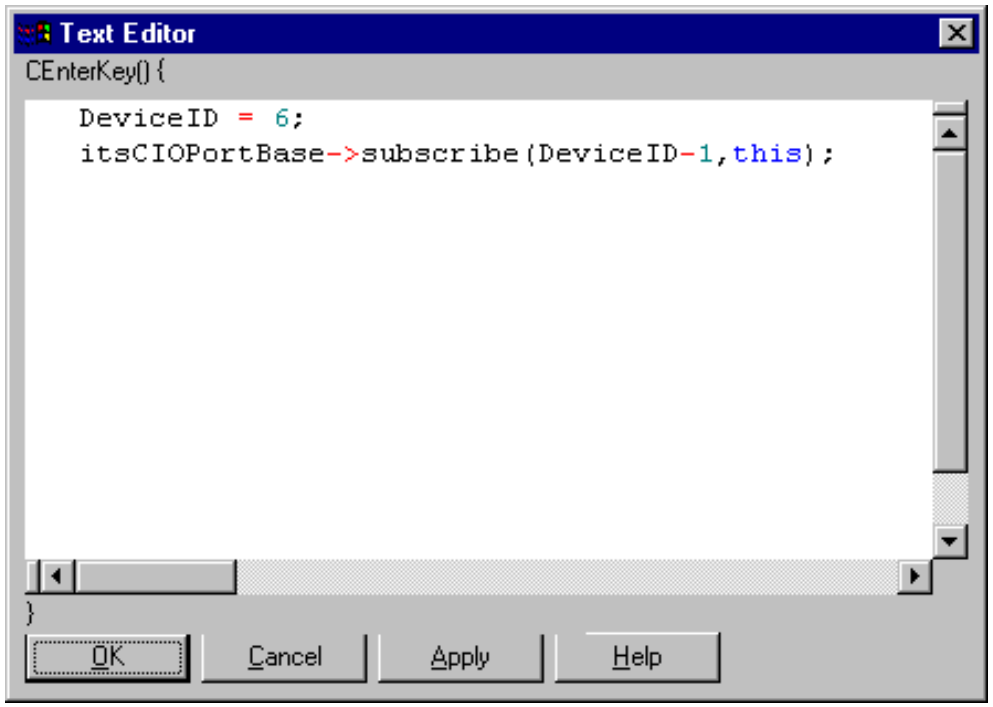
Für die zentrale Ablaufkontrolle ist im Softwaremodell zusätzlich die Klasse CController eingeführt.

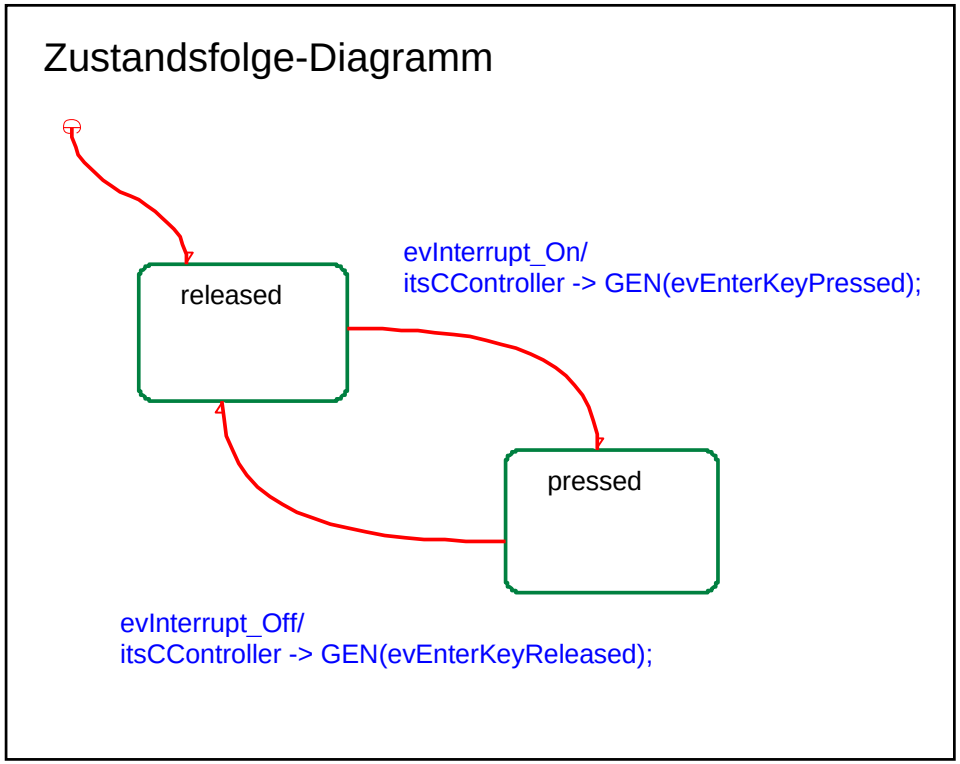
Klassen-Diagramm



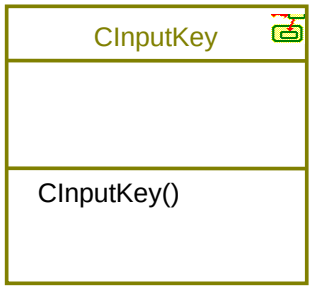
Konstruktor und Zustandsfolge-Diagramm

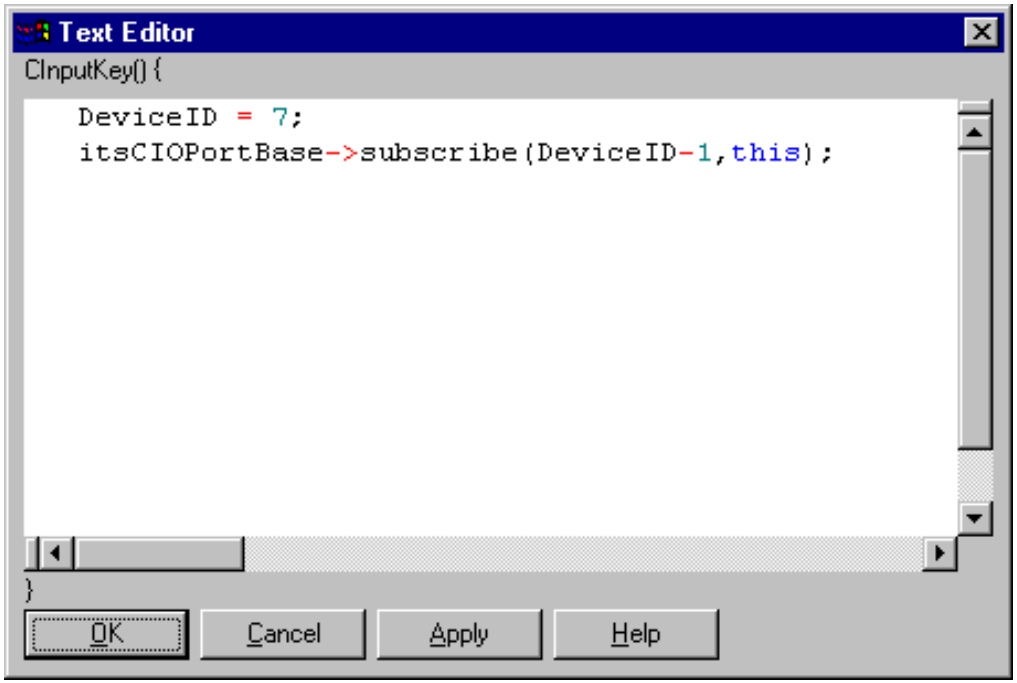


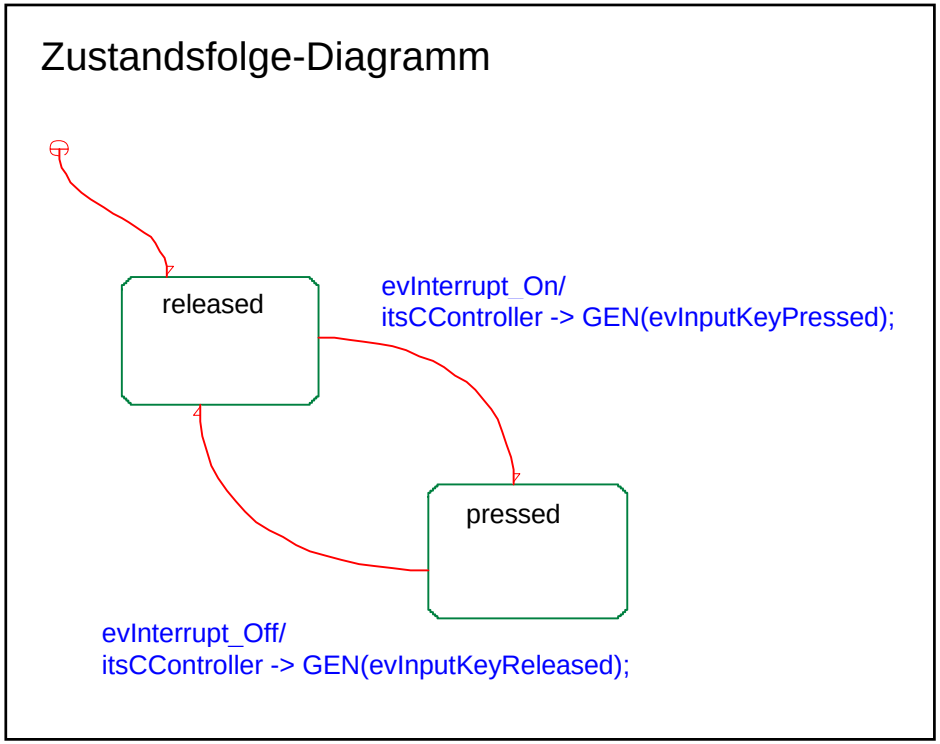




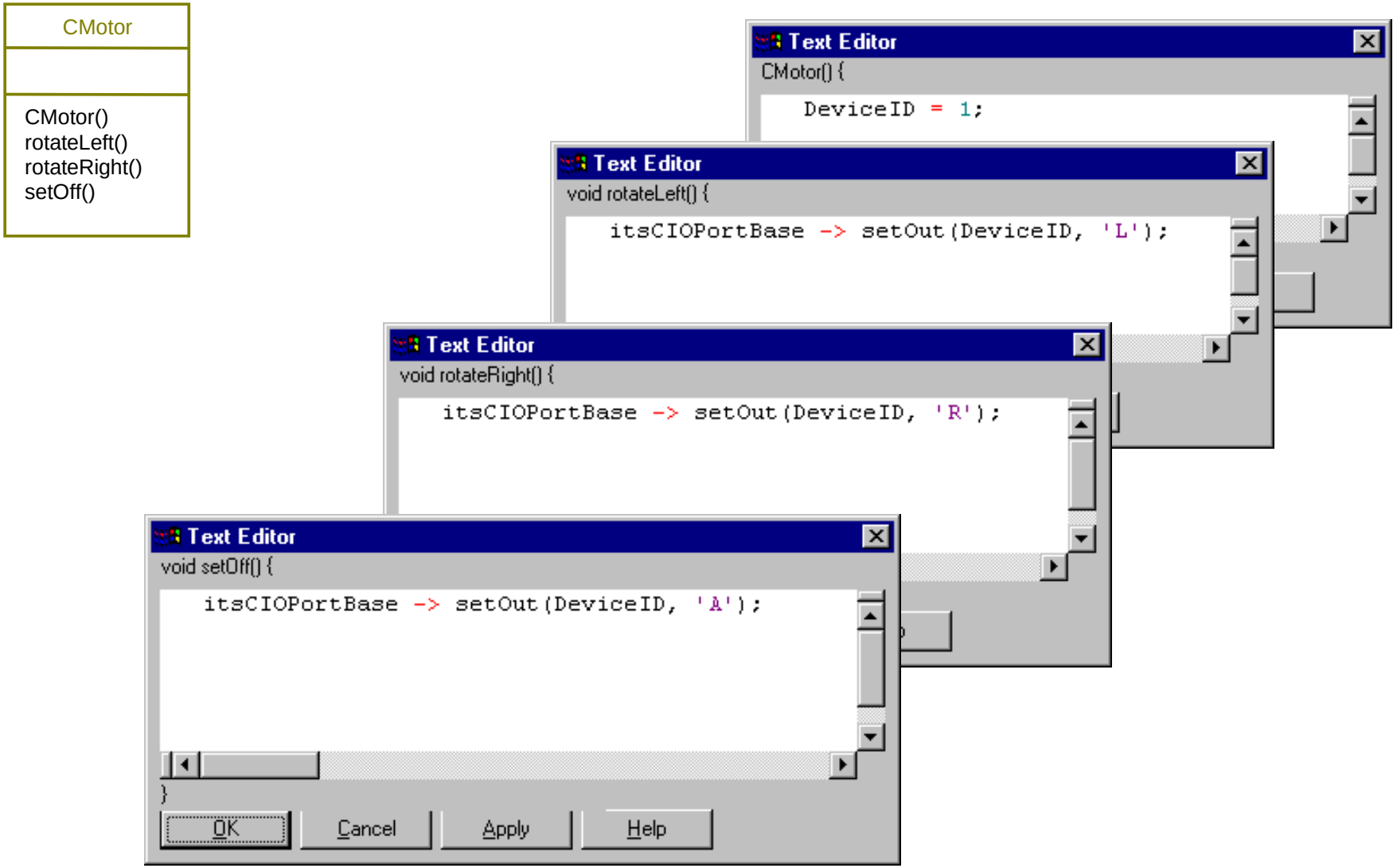
Konstruktor und Zustandsfolge-Diagramm



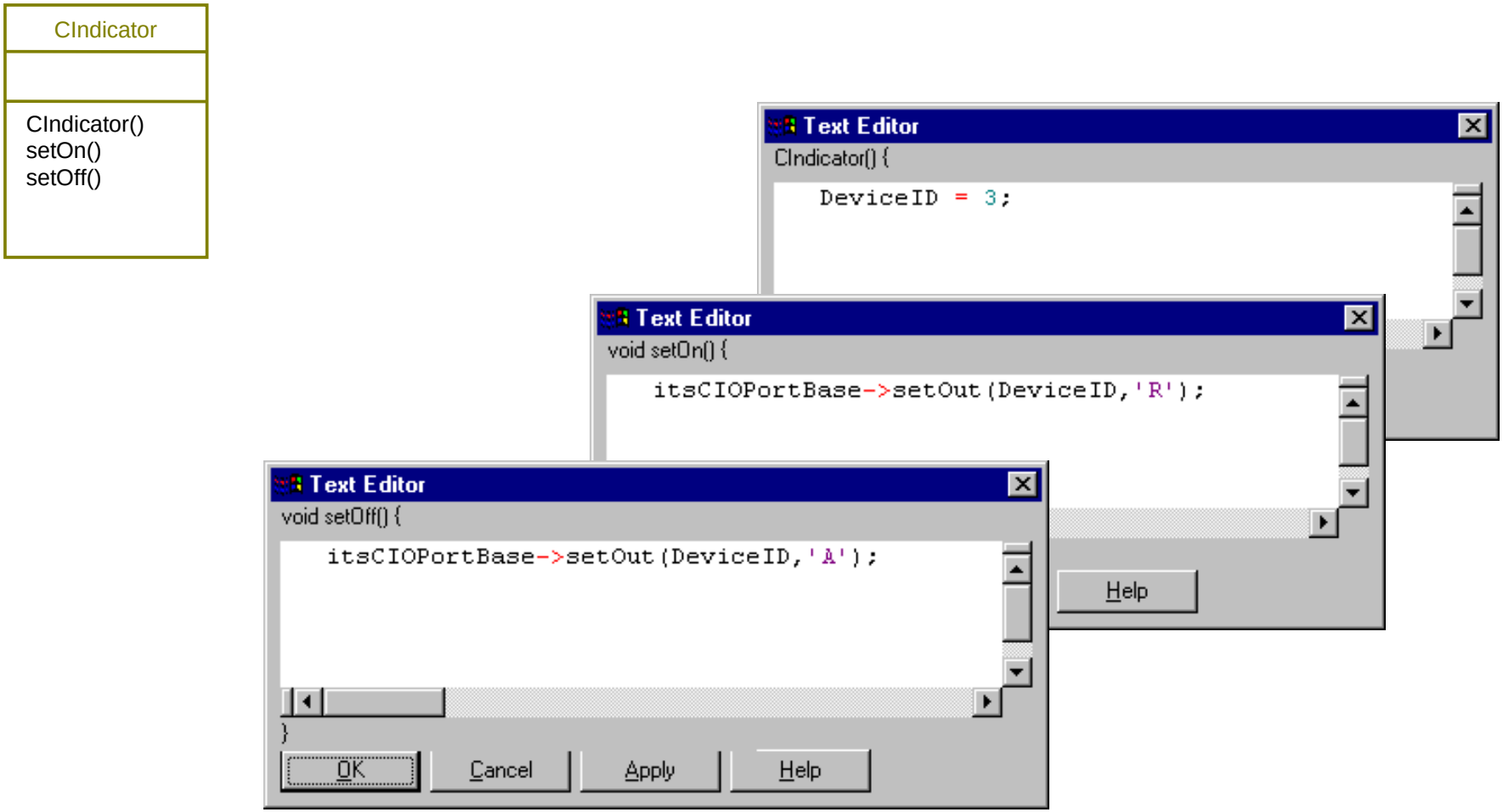




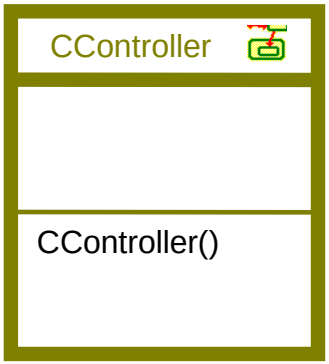
Konstruktor und Methoden rotateLeft(), rotateRight(), setOff()

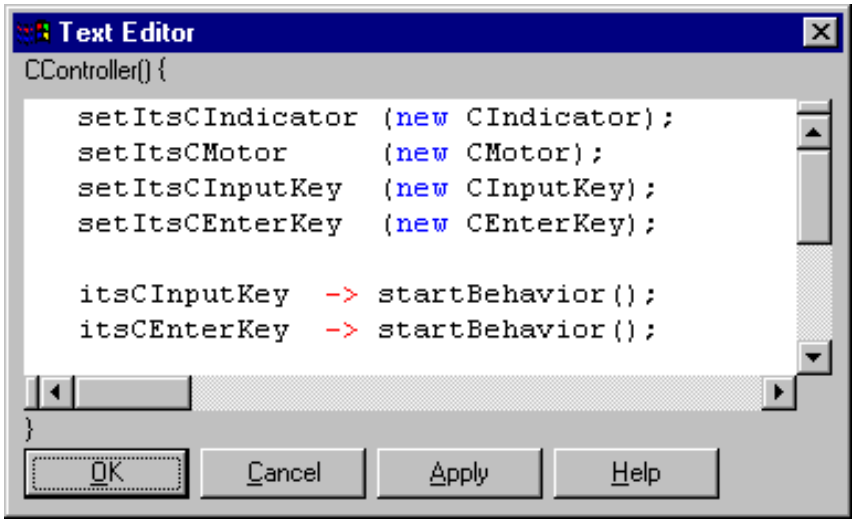


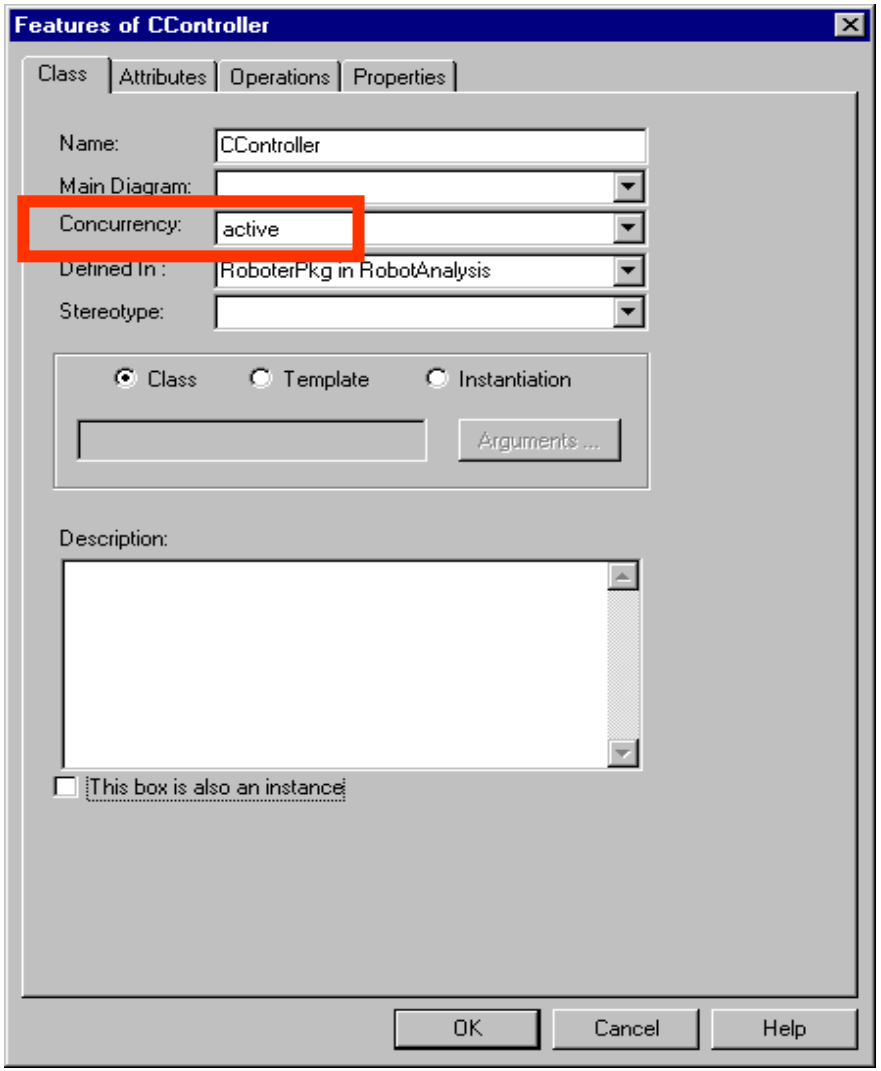
Konstruktor und Methoden setOn(), setOff()



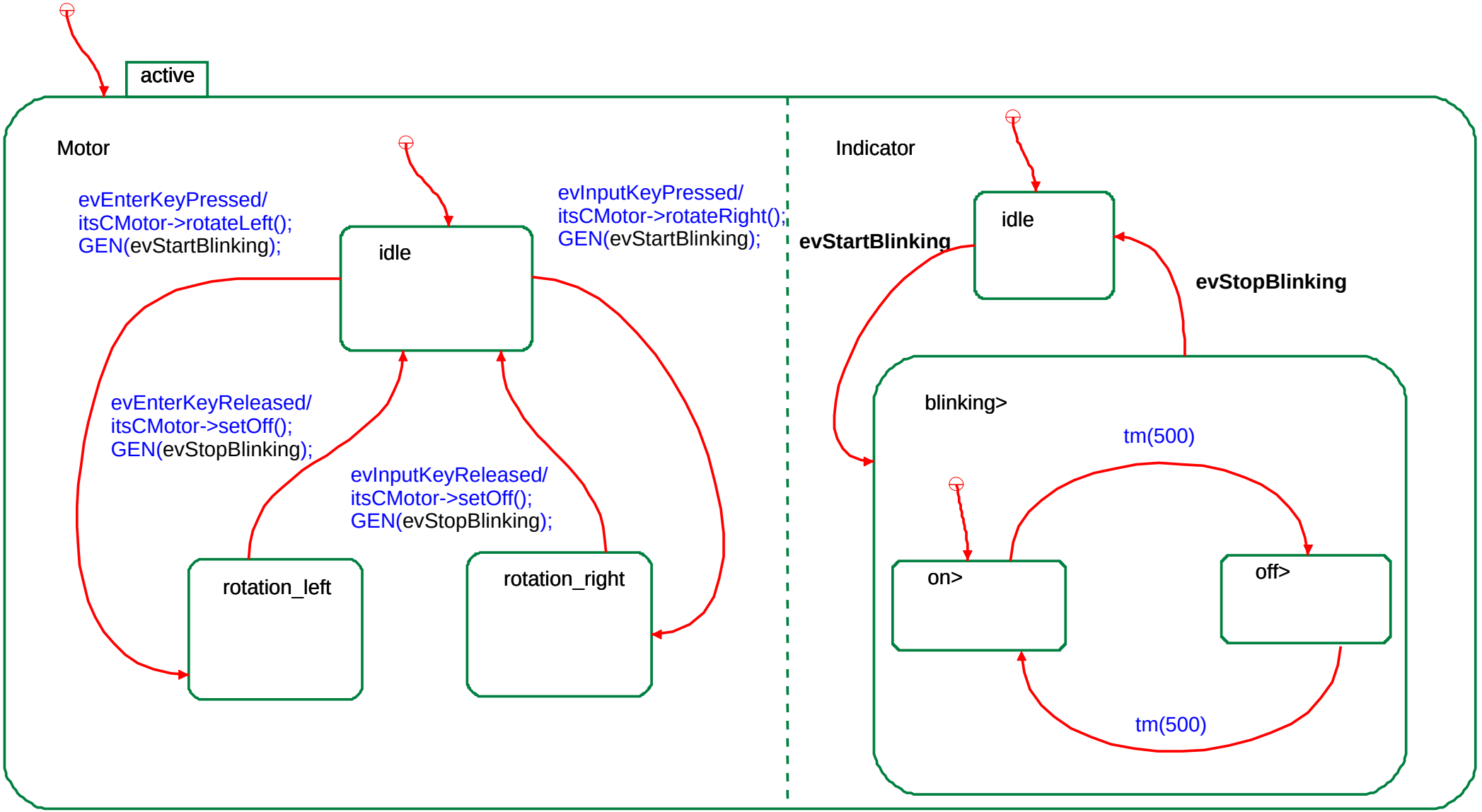
Konstruktor und Definition als aktive Klasse (Thread)



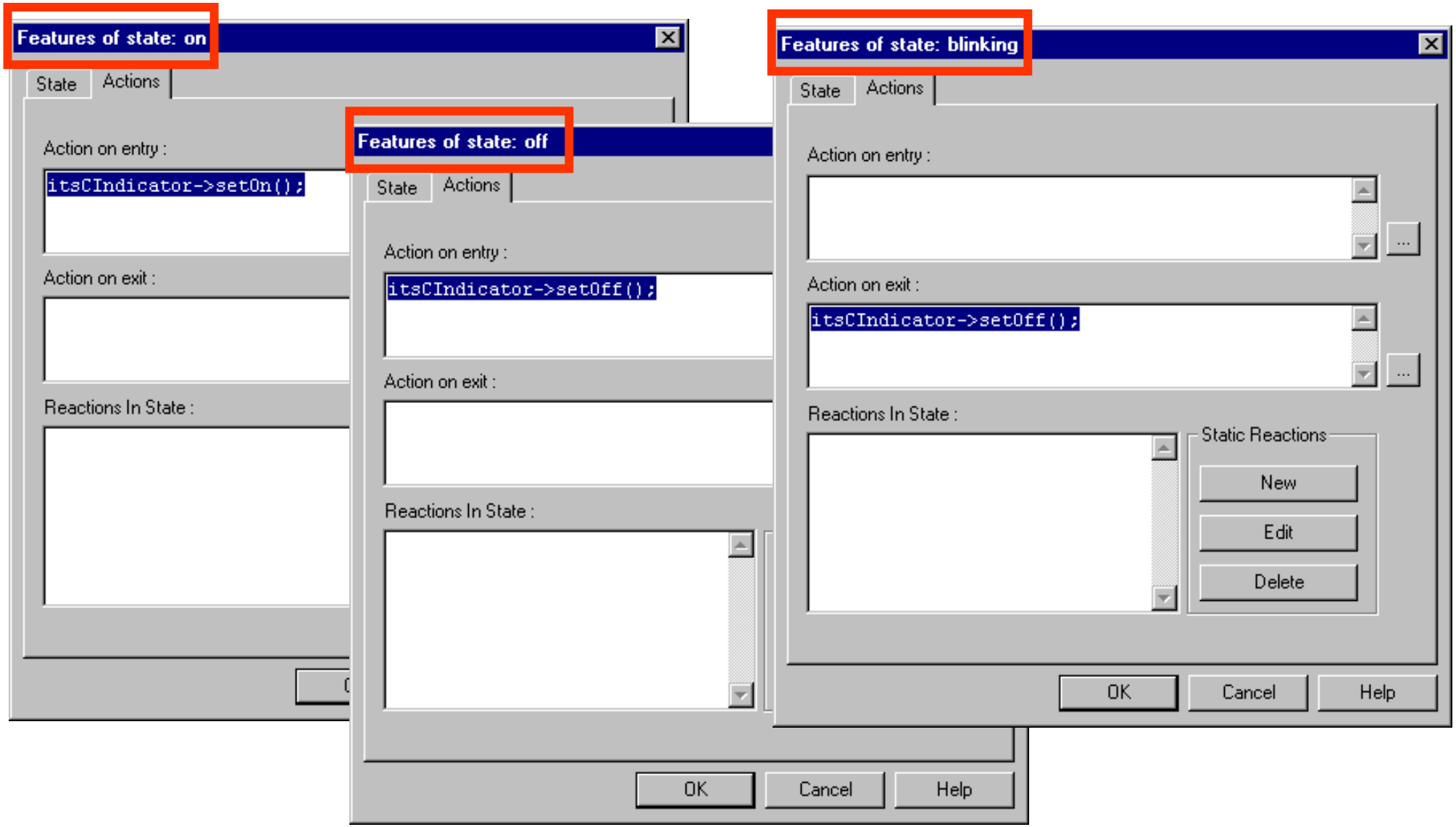




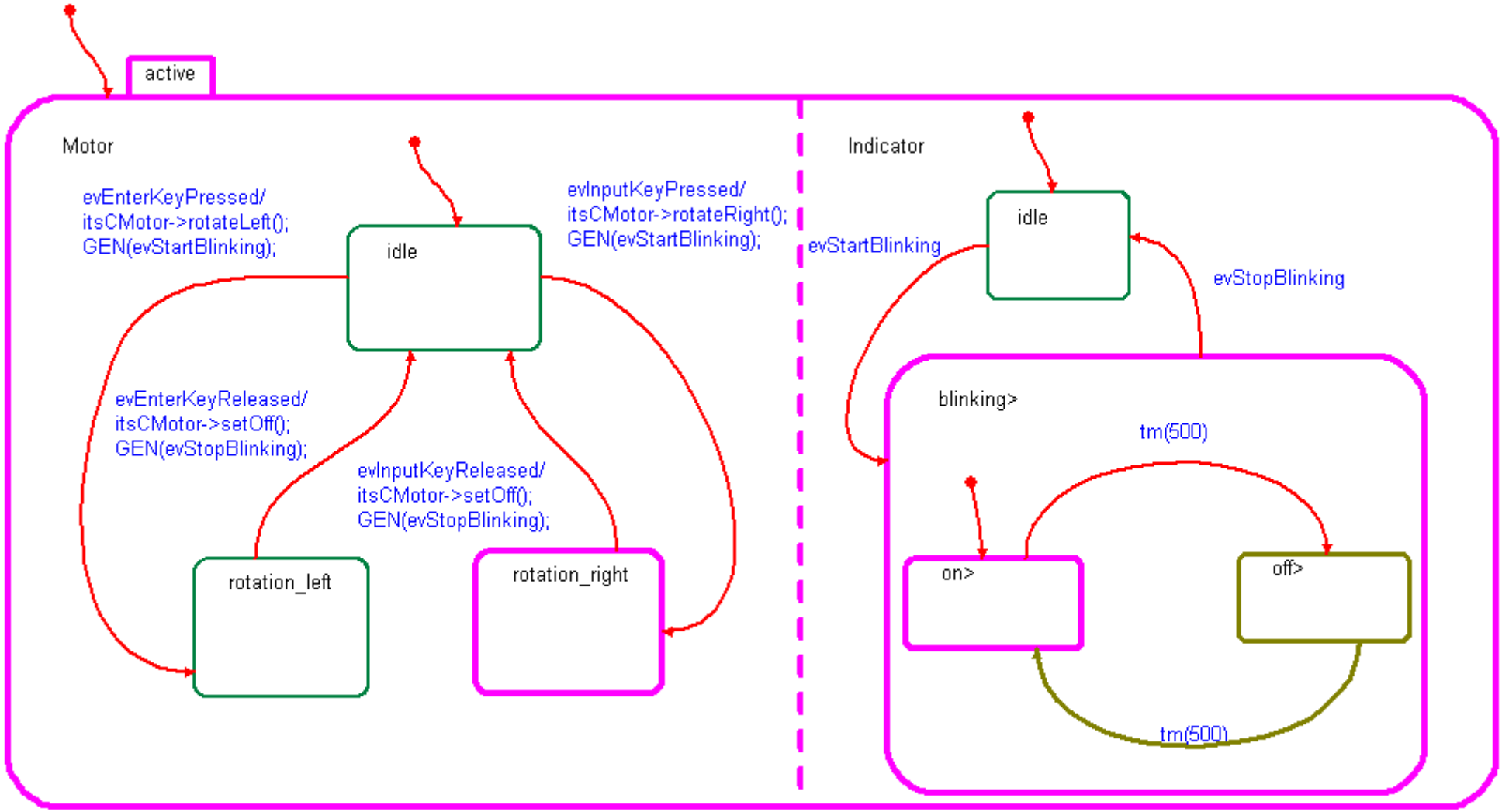
Zustandfolge-Diagramm



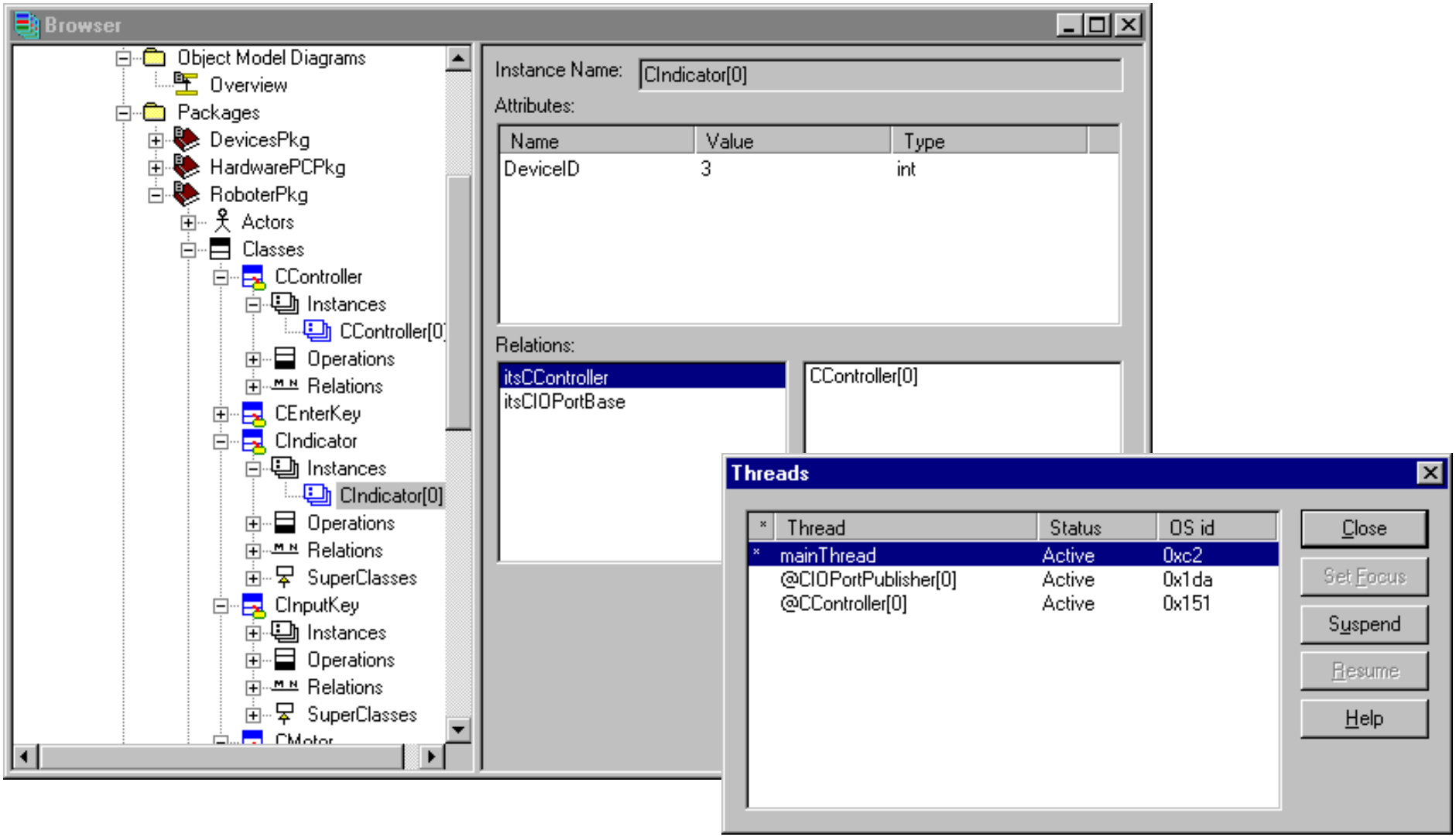
Aktionen beim Verlassen und Eintreten in die entsprechenden Zustände (on, off, blinking)



Animiertes Zustandsfolge-Diagramm



Ansicht der Attribute und Übersicht der Threads



Wie wird ein UML-Case-Tool in der Praxis eingesetzt ?

1. Requirement-Analyse mit dem Use-Case-Diagramm
2. Beschreibung der statischen Struktur mit dem Klassen-Diagramm
3. Beschreibung des dynamischen Verhaltens mit dem Zustandsfolge-Diagramm
4. Konfiguration und Codegenerierung
5. Ausführung, Verifikation und Debugging des Modells mittels animiertem Zustandsfolge-Diagramm

Welche UML-Diagramme sind für die Codegenerierung notwendig ?

Statische Struktur:	Klassen-Diagramm
Dynamisches Verhalten:	Zustandsfolge-Diagramm

Wie sieht die Unterstützung für Verifikation und Debugging aus ?

- Durch animierte Zustandsfolge-Diagramme kann der Programmablauf überprüft werden. Ereignisse können simuliert und ganze Ereignislisten (Script) abgearbeitet werden.
- Auf Objektbasis können Breakpoints gesetzt werden, und alle Objekte mit ihren Attributen und Zuständen sowie die laufenden Threads sind sichtbar.