Roboter-OS mit FORTH

April 14, 2007

Gerd Franzkowiak

FORTH als Programmiersprache und als Betriebssystem

Contents

1 Einleitung

Roboter sind eine ideale Umgebung für Disziplinen wie

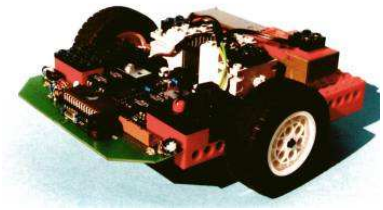
- Elektronik,
- Software,
- Sensorik,
- Mechanik,
- Mikrocomputertechnik.

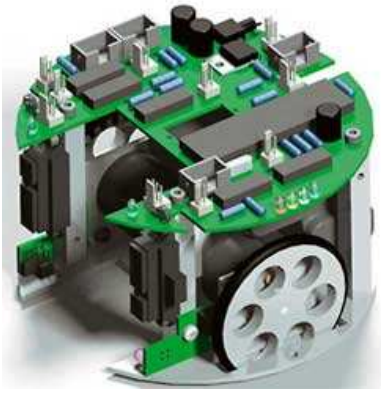
Genau diese technische Symbiose und die Art der Automatisierung ist eine ideale Umgebung für FORTH. Das hat mich veranlaßt, ein ***FORTH-Roboter-Projekt als Freie Software*** z.B. unter den Bedingungen der GPL anzuregen. Es könnte ein Aushängeschild für FORTH werden.

2 Roboter-Übersicht

2.1 Roboter-Varianten

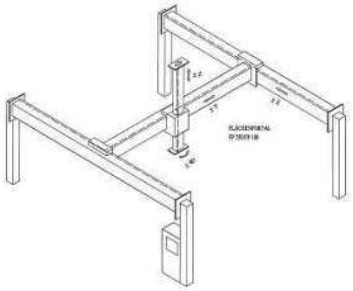
2.1.1 Hobby, lern und akademische Varianten



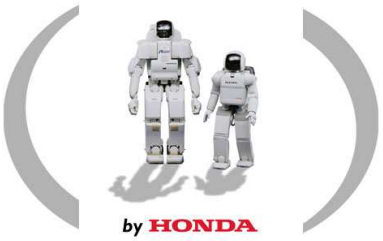


2.1.2 Knickarm-Roboter (der Klassiker)

2.1.3 Flächenportal-Roboter (Favorit für den Einstieg)



2.1.4 Humanoiden (die Zukunft, auch für üble Typen)



3 Grundgedanken

3.1 Die Quelle der Gedanken

- Für die industriellen Varianten der Roboter-Technik existiert kein Programmierstandard !
- Jeder Hersteller verwendet seine eigen Programmiersprache.

  Erste Tendenzen sind zu erkennen, daß Hersteller sich in Richtung des Standards der "Speicherprogrammierbaren Steuerungen" (SPS) bewegen. Der Standard dafür ist unter der IEC 61131-3 beschrieben und umfaßt mehrere Methoden der Programmierung.

  Siehe VD 2006-4: *Leserbriefe und Meldungen "Die Anwendung – Automatisierung"*.
- Meine Idee dazu war und ist, die Methodik "Strukturierter Text / *ST*" als eine Art Schale um FORTH zu legen und somit eine Brücke für Nutzer aus der Industrie bauen.

  Genau diese Nutzer können dann in einer gewohnten Umgebung programmieren und mit einer integrierten Entwicklungsumgebung die Sourcen auf eine Steuerung schieben. Dort, auf der Steuerung, übernimmt ein Tool z.B. entsprechend Anton Ertls Paket "Gray" die Aufgabe, diese IEC 61131-Sourcen in FORTH zu wandeln und in die RunTime-Umgebung einzubinden.
- Die Verfahrensweise hat aus meiner Sicht drei Vorteile:

1. FORTH könnte sich in einem Wachstumsmarkt der Automatisierung etablieren,
2. der erfahrene FORTH-Anwender kann den Roboter voll ausreizen und
3. der "alteingesessene" Programmierer der Automatisierung findet seine gewohnte Umgebung und könnte so ganz nebenbei sogar Berührung mit FORTH bekommen.

3.2 Die Sicht der Automatisierung

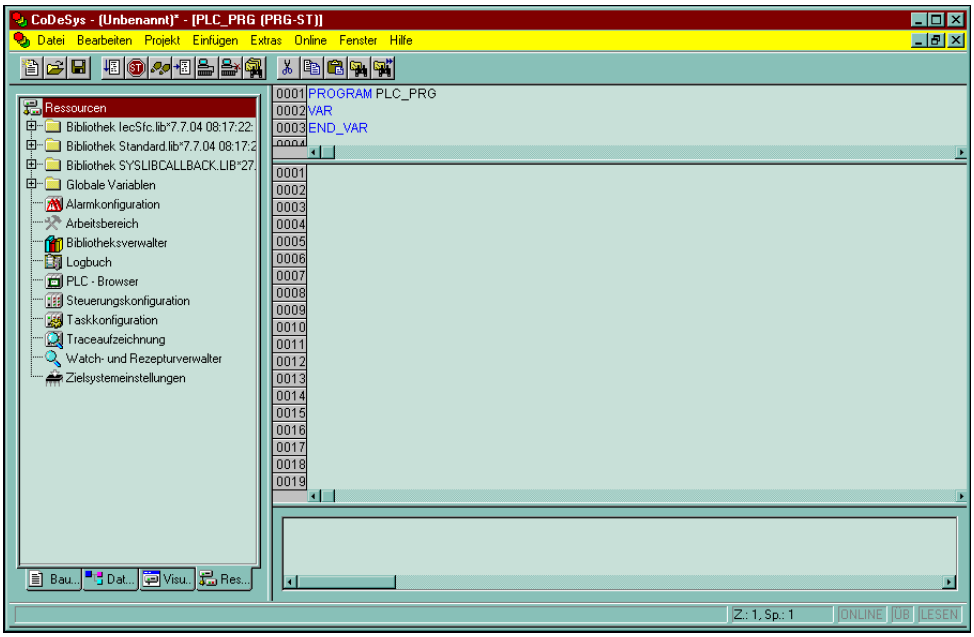
3.2.1 Entwicklungsumgebung

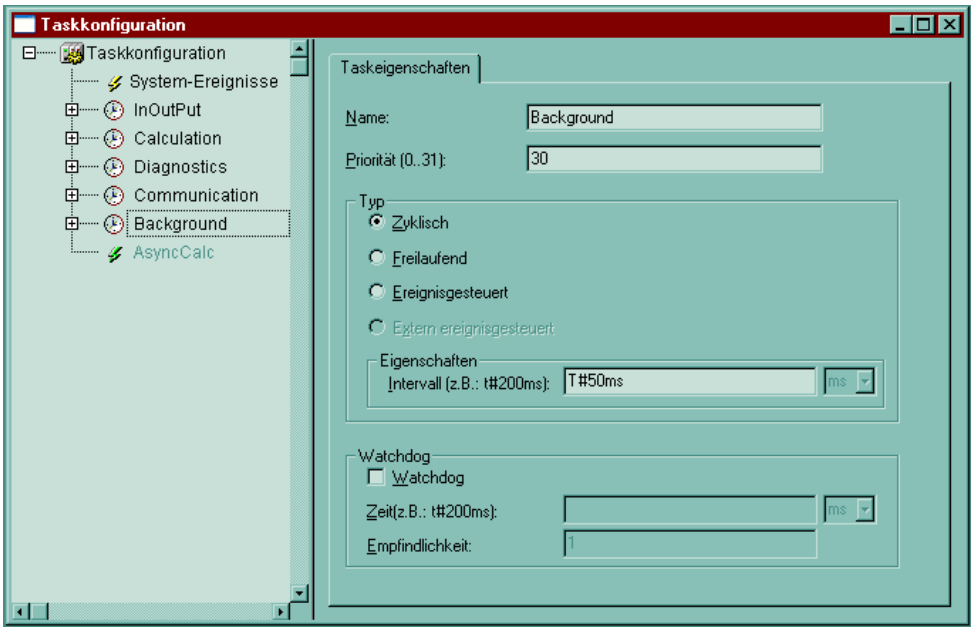
- Als Beispiel stelle ich hier eine Ansicht der *CoDeSys* Entwicklungsumgebung entsprechend IEC 61131 vor. Sie ist ein Produkt der Firma

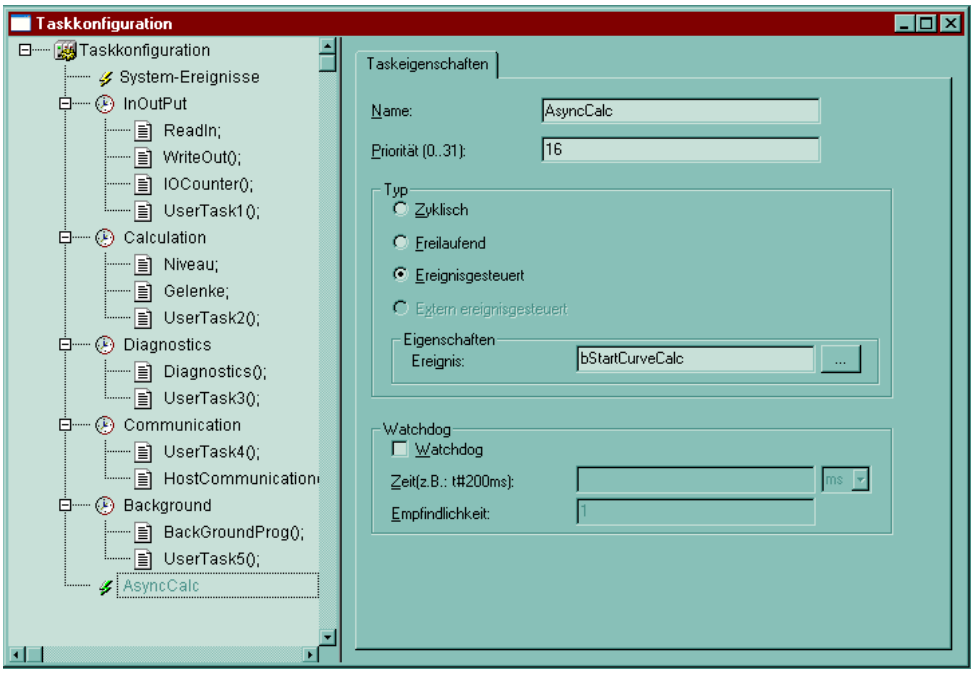
  3S - Smart Software Solutions GmbH

  Memminger Straße 151

  87439 Kempten

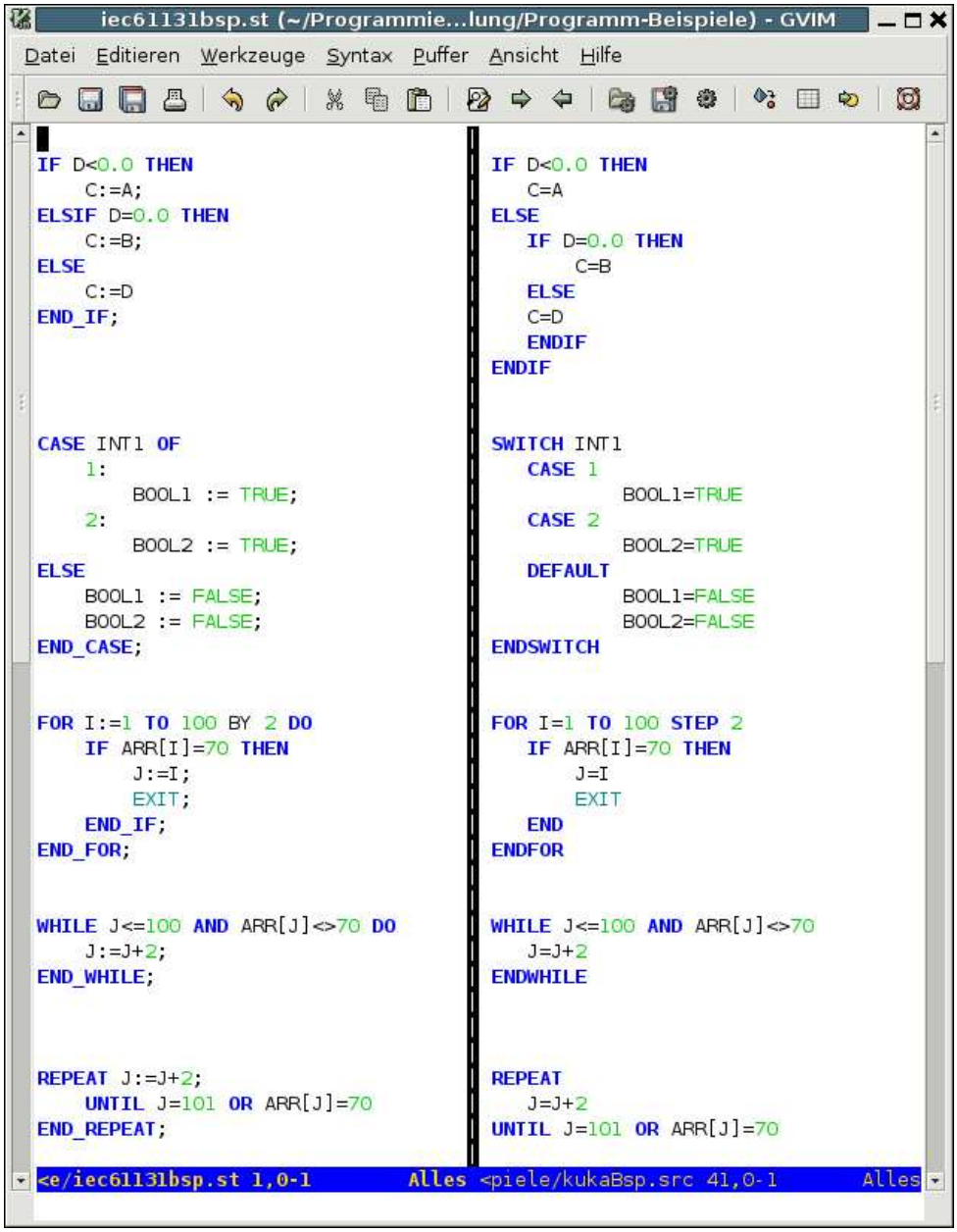



Taskkonfiguration
Taskkonfiguration
System-Ereignisse
InOutPut
Calculation
Diagnostics
Communication
Background
AsyncCalc
Taskeigenschaften
Name:
Background
Priorität (0..31):
30
Typ
Zyklisch
Freilaufend
Ereignisgesteuert
Extern ereignisgesteuert
Eigenschaften
Intervall (z.B.: t#200ms):
T#50ms
ms
Watchdog
Watchdog
Zeit(z.B.: t#200ms):
ms
Empfindlichkeit:
1


Taskkonfiguration
Taskkonfiguration
System-Ereignisse
InOutPut
ReadIn;
WriteOut();
IOCounter();
UserTask1();
Calculation
Niveau;
Gelenke;
UserTask2();
Diagnostics
Diagnostics();
UserTask3();
Communication
UserTask4();
HostCommunication
Background
BackGroundProg();
UserTask5();
AsyncCalc
Taskeigenschaften
Name:
AsyncCalc
Priorität (0..31):
16
Typ
Zyklisch
Freilaufend
Ereignisgesteuert
Extern ereignisgesteuert
Eigenschaften
Ereignis:
bStartCurveCalc
...
Watchdog
Watchdog
Zeit(z.B.: t#200ms):
ms
Empfindlichkeit:
1

3.2.2 Das Aussehen der Programme

IEC 61131 KUKA Robot Language

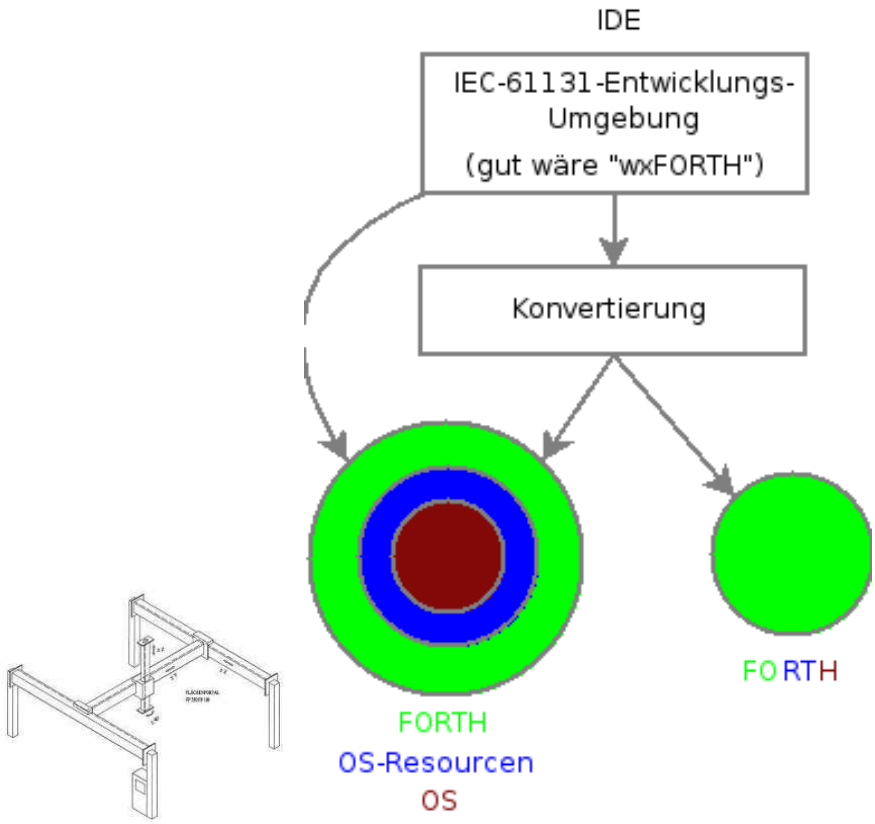


4 Vorschlag zur Vorgehensweise

4.1 Überblick

Für einen ***BOTTOM-UP*** Entwicklungsweg sollte FORTH für den Einsatz auf 2 Wegen zum Einsatz kommen.

1. FORTH auf Linux, um alle Tools und Möglichkeiten für die Entwicklung zu haben
2. FORTH stand alone für das High End und absolute Portabilität sowie Sicherheit und Zertifizierung.



4.2 Echtzeit / Multitasking

Es zeichnet sich ab, daß es relativ schwierig zu realisieren ist, FORTH in ein Realtime-OS einzufügen. In diesem Zusammenhang stellen sich z.B. die Fragen:

1. *Eine* FORTH-Umgebung aufsplitten für verschiedene Prozesse / Tasks oder

2. *Mehrere* FORTH-Umgebungen in je einem Prozeß / einer Tasks des Betriebsystems ?

   Für die entgültige Robotersteuerung sind Tasks wie:
   Antriebs-Regelung,
   Bahnplanung / Steuerung,
   Feldbus-Kommunikation,
   Bediener-Kommunikation und
   nicht zuletzt für asynchrone Ereignisse
   erforderlich.

Aus dem Bild der CodeSys-Entwicklungsumgebung `"Task-Konfiguration"`3.2.1 war zu erkennen, daß unterschiedliche und mehrere Programme einer gewünschten Task zugeordne werden müssen.

> *Ich denke, diese Aufgaben komplett in FORTH zu lösen wäre wohl ein erster, bestimmt auch sehr interessanter Schritt.*

4.2.1 Harte Echtzeit

Garantierte anwendungsabhängige worst-case Reaktionszeiten (nicht unbedingt kurz, aber sehr deterministisch)

- Realisierung mit der Kernel-Erweiterung Xinomai für Linux
- FORTH stand alone ?

4.2.2 Weiche Echtzeit

Rechtzeitigkeit wird benötigt, aber Scheitern bleibt ohne schwerwiegende Folgen (evtl. Qualitätsverluste)

- Realisierung mit dem Kernel-Patch von Ingo Molnar und Andrew Morton für Linux
- FORTH stand alone

5 Mögliches Time Scheduling

Der gesamte Zyklus der Entwicklung sollte sich in 3 Phasen gliedern, die auch teilweise parallel laufen könnten.

5.1 Echtzeit-Multitasking-System

- Erstellen des Konzeptes, welcher Weg beschritten werden soll, um Multitasking-Fähigkeit für *FORTH* auf Robotern und in der Automatisierung zu schaffen,
- Möglichkeiten für erweiterbare Prozeß- bzw. Task-Listen, wie im Bild oben gezeigt, schaffen,
- Prinzipien der Interprozeßkommunikation festlegen,
- Debugging-Umgebung für Inbetriebnahme erstellen (Inbetriebnehmer der Roboterwelt kennen und lieben den Step-Betrieb).

5.1.1 Realisierung der Multitasking- und Echtzeit-Fähigkeiten von FORTH auf Linux

Resourcen des Betriebssystemes, wie z.B. die Interprozess-Kommunikation, nutzen.

5.1.2 Realisierung der Multitasking- und Echtzeit-Fähigkeiten der Stand-Alone-FORTH-Umgebung

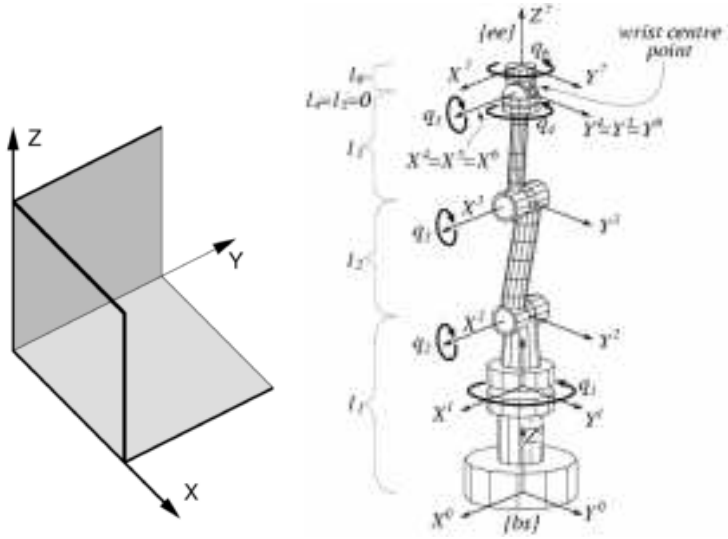
Art und Weise des System-Konzeptes festlegen.

5.2 Steuer- und Regelsystem für einen Roboter

Diese Aufgaben können mit Sicherheit aus den Erfahrungen des Linux-CNC-Systems EMC Enhanced Machine Controller `http://www.linuxcnc.org` schöpfen, da viele Parallelen in Fragen der Transformationen vorhanden sind.

Beispiel:

Achswinkel von 6 Achsen müssen vor- und rückwärts transformiert werden um jederzeit die karthesiche Position der Werkzeugspitze zu kennen. Für eine Bahnplanung muß dies gegenwärtig ca. alle 6 ms erfolgen, auch für unterschiedliche Geometrien (Knickarm-Roboter, Portal-Roboter).



5.3 Schaffung eines Automatisierungssystems mit integrierter Entwicklungsumgebung

Um den Roboter-Kern muß eine Umgebung geschaffen werden, welche eine Akzeptans nach außen bringt. Dies entspricht meinen vorangegangenen Äußerungen zum Einsatz der IEC 61131 und einer möglichen Entwicklungsumgebung, z.B. mit Gray und einer wxWidgets-Anknüpfung für FORTH.