AL - EIN TEXTUELLES PROGRAMMIER-SYSTEM FÜR INDUSTRIEROBOTER

Dipl.-Inform. C. Blume

Universität Karlsruhe
Lehrstuhl für Planungs- und Programmiertechniken für Prozeßrechner

3. Juli 1980

Spezielle Probleme der Roboterprogrammierung

Die Programmierung von Industrierobotern wirft einige spezielle Probleme auf, die in der "normalen" Datenverarbeitung kaum vorkommen. So muß der Programmierer in irgendeiner Form dem System mitteilen, wie der Roboter im dreidimensionalen Raum bewegt werden und welche Orientierung der am Roboterarm befindliche Greifer einnehmen soll. Für einen Menschen ist es i.a. unmöglich, Operationen im dreidimensionalen Arbeitsraum des Roboters mittels einer Folge exakter Positions- und Orientierungsangaben zu beschreiben. Sieht man einmal von den exakten geometrischen Angaben ab, so stellt sich bei komplexeren Aufgaben immer noch das Problem, wie der Programmierer in einfacher, seiner Vorstellungswelt angepaßter Weise die räumlichen Operationen des Roboters formulieren kann. Damit hängt ein weiteres Problem zusammen: die möglichst vollständige Beschreibung der externen Umwelt des Roboters, d.h. der Arbeitssphäre mit den zu manipulierenden Objekten. Diese Daten ermöglichen erst eine abstraktere, funktionsorientierte Programmierung. Außerdem bilden sie zusammen mit Sensormeldungen die Grundlage für eine Anpassung an die externe Umwelt durch das Laufzeitsystem sowie für eine eventuelle Fehlererkennung und -korrektur im Funktionsablauf. Die Sensorintegration soll Reaktionen auf Veränderungen der realen Umwelt (des Roboters) ermöglichen, die unbeabsichtigt, irreversibel oder auch dynamisch (z.B. die dauernde Eigenbewegung eines Objekts betreffend) sein können.

Methoden der Roboterprogrammierung

Wegen der oben beschriebenen Problematik verwendet man unterschiedliche, von der "normalen" Programmierung abweichende Verfahren, wobei das "freie Definieren" einer Bewegungsbahn des Roboters mit ihren Anfangs-, Zwischen- und Endpunkten im Vordergrund steht. Außerdem muß die Orientierung eines Greifers oder Werkzeugs am Roboter beachtet werden, d.h. die relative Lage bezüglich eines Bewegungsbahnpunktes. Grundsätzlich lassen sich vier Programmierverfahren für Industrieroboter unterscheiden:

a) manuelle Programmierung
b) Folgeprogrammierung
c) Teach-in-Verfahren
d) textuelles Programmieren

Die manuelle Programmierung erfolgt durch das physikalische Einrichten von Haltepunkten auf jeder Bewegungsachse des Roboters mittels Anschlägen, Luftschläuchen, Steckerfeld u.a.. Hierzu ist i.a. kein Rechner erforderlich und die Programmierung beschränkt sich auf Roboterbewegungen.

Die Folgeprogrammierung basiert auf dem Prinzip, daß der Programmierer den Roboter auf seine Handbewegungen im Arbeitsraum folgen läßt. Dabei werden die so angefahrenen Positionen und Orientierungen statisch (d.h. auf Anforderung) oder dynamisch (d.h. nach einer Zykluszeit) abgefragt und abgespeichert. Der Roboter kann dann die gespeicherte Bewegungsfolge reproduzieren. Das Verfolgen der Handbewegung des Programmierers kann aufgrund taktiler oder optischer Sensoren erfolgen.

Auch beim Teach-in-Verfahren bewegt der Programmierer den Roboter in die gewünschte Position und Orientierung, jedoch nicht direkt, sondern mit Hilfe einer Programmiereinheit. Diese besteht aus einem Kasten mit verschiedenen Funktionstasten und Drehknöpfen, mit deren Hilfe das Anfahren eines Bahnpunktes gesteuert werden kann. Außerdem kann die Programmiereinheit Funktionstasten enthalten für das Öffnen/Schließen der Hand, Geschwindigkeitsangaben, Steuerungsart der Bewegung (Punkt-zu-Punkt oder Bahnsteuerung), Wartezeiten, Unterbrechungen und sogar für das Programmieren einfacher Schleifen und Sprünge im so erstellten Programm.

Von einem ausgebauten Teach-in-Verfahren ist es dann nur noch ein Schritt bis zur textuellen Programmierung, da der Anwender auch beim Teach-in mit abstrakten Symbolen umgehen muß. Sie werden nur nicht wie bei der textuellen Programmierung in Form von Zeichenketten dargestellt, sondern dem System durch Drücken von Tasten mitgeteilt.

Vergleich der textuellen Programmierung mit dem Teach-in-Verfahren

Daraus resultieren folgende Vorteile der textuellen Programmierung gegenüber den anderen nicht-textuellen Verfahren:

- Das Programm ist für den Programmierer und alle anderen Benutzer lesbar, daher mitteilbar.

- Der Programmtext kann in lesbarer Form gespeichert werden (Dokumentation).

- Das Programm kann (auch von anderen Programmierern) korrigiert und verändert werden (Wartung).

- Der Programmtext kann ohne Roboter erstellt und übersetzt werden.

- Das Programm kann Variablen enthalten (d.h. die Daten sind beim Programmieren nur in ihrer Struktur, nicht ihrem Wert nach festgelegt).

- Im Programm sind komplexe Verzweigungsangaben möglich (d.h. der Programmablauf kann sich differenziert an äußere Gegebenheiten anpassen).

- Das Programm kann hierarchisch modular aufgebaut werden (d.h. es sind Unterprogramme und Prozeduren möglich).

- Es sind geschachtelte Befehle und Aufrufe sowie rekursive Prozeduren möglich.

Die obige Aufstellung läßt drei prinzipielle Schwachstellen des Teach-in-Verfahrens erkennen:

1) fehlende Mitteilbarkeit und Dokumentation

2) wenig differenzierter Programmablauf

3) keine Datenmanipulation

Daher wird die textuelle Programmierung vor allem dann benötigt, wenn

a) der Roboter mit Sensoren ausgestattet ist, da entsprechend der Sensordaten der Programmablauf verändert wird (Punkt 2)

b) auf externe Daten zugegriffen wird (z.B. bei Integration in ein CAD-oder CAM-System)

c) das Laufzeitsystem für eine automatische Fehlererkennung und -korrektur ein Umweltmodell benötigt (Punkt 3).

Allerdings hat bis jetzt auch die textuelle Programmierung eine Schwachstelle, nämlich die Schwierigkeit, Bewegungspunkte zu definieren. Der Programmierer kann zwar prinzipiell alle Positionen und Orientierungen innerhalb eines Koordinatensystems explizit angeben, jedoch ist dies i.a. nicht möglich, da kein Mensch eine Bewegungsfolge im dreidimensionalen Raum nur nach der Vorstellung millimetergenau in Koordinaten beschreiben kann. Daher enthält vorläufig jedes textuelle Programmiersystem eine Teach-in-Komponente, die zum Aufnehmen der Bewegungspunkte mit Hilfe des Roboters dient. Der Programmierer fungiert dabei als Regler, der die Abweichung vom (realen) Zielpunkt durch die Augen "mißt" und den Roboterarm entsprechend steuert.

Aufbau der Sprache AL

Die Sprache AL (Assembly Language) der Stanford University wurde nicht als roboterspezifische Sprache neu geschaffen, sondern von ALGOL abgeleitet. Dabei wurden

- Blockstruktur
- Typendeklaration
- Verzweigungen (IF THEN)
- Schleifen (FOR, WHILE, UNTIL)
- Auswahl (CASE OF)
- arithmetische und boolesche Ausdrücke
- höhere mathematische Routinen (SQRT, SIN, TAN, LOG, u.a.)
- Prozeduren
- Textein/ausgabe

von ALGOL übernommen. Neu definiert wurden die Datentypen:

Programmierung in AL

Die Anfangs-, Zwischen- und Endposition einer Roboterbewegung werden in AL durch die Frames beschrieben. Ein Frame besteht aus einer Vektor- und einer Rotationsangabe bezüglich eines Standardkoordinatensystems; es stellt selbst ein kartesisches Koordinatensystem dar. Der Vektor gibt die Verschiebung des Ursprungs des Koordinatensystems an, die Rotationsangaben beziehen sich auf Drehungen um die Achse des Standardkoordinatensystems.

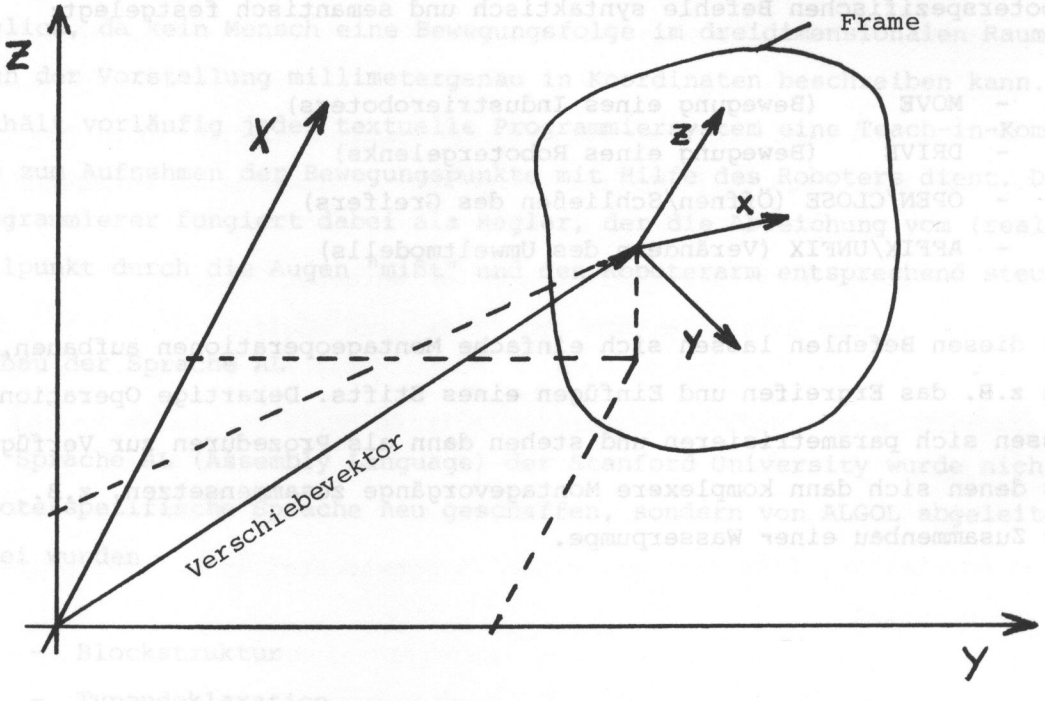

Bild 1: Geometrische Beschreibung eines Frames

Man kann nun Relationen zwischen Frames herstellen, indem man den Übergang von einem Frame in ein anderes wieder mittels Verschiebevektor und Rotationsangaben beschreibt. Eine solche Relation repräsentiert in AL der Datentyp TRANS.

Das Definieren eines Frames kann durch explizite Koordinatenangaben für den Verschiebevektor und Rotationsangaben bezüglich Rotationswinkel und -achse erfolgen. Im allgemeinen wird der Programmierer jedoch den interaktiven Modus des Programmiersystems benutzen, der es erlaubt, mittels des Teach-in-

Verfahrens eine Position und Orientierung mit dem Roboter anzufahren. Anschließend werden über Lagesensoren die Roboterkoordinaten eingelesen, in kartesische Koordinaten umgerechnet, sowie die so gewonnenen Werte einem vom Anwender am Terminal eingegebenen Framenamen zugeordnet. Die Frames werden in einer gesonderten Liste gesammelt und stehen zum Programmablauf zur Verfügung. Der Programmierer erstellt also off-line auf einem Hintergrundrechner sein Programm und definiert später on-line die entsprechenden Framewerte, wobei die Zuordnung der Frames in der Liste zu denen im Programmtext über symbolische Namen erfolgt.

Eine Bewegung des Roboters wird in AL durch den MOVE-Befehl programmiert. Die Geometrie der Bewegung, d.h. die Bewegungsbahn und die an einzelnen Positionen einzunehmende Orientierung, wird mit Hilfe der Frames beschrieben. Die Z-Achse des Frame-Koordinatensystems gibt nämlich die Richtung der letzten Roboterachse an, die Y-Achse geht durch beide Greifpunkte des Greifers, und die X-Achse steht senkrecht auf beiden.

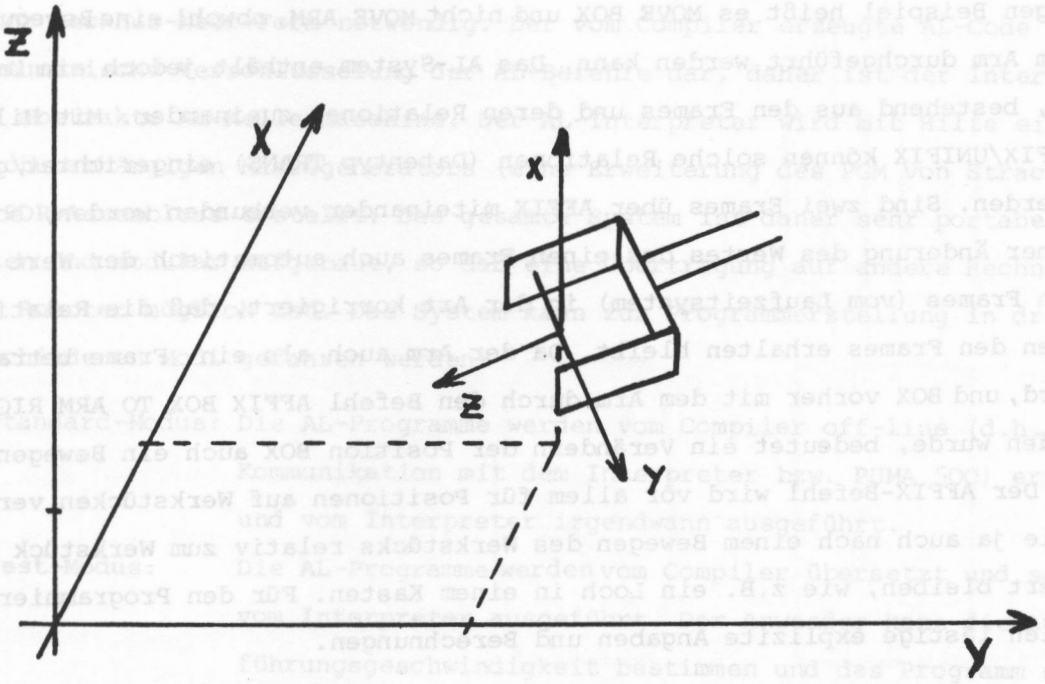

Bild 2: Zusammenhang zwischen Frame und Position und Orientierung des Greifers

Außerdem kann der MOVE-Befehl noch Angaben über die Geschwindigkeit beim Durchfahren einer Position und über die Bewegungszeit enthalten. Soll die Bewegung oder der Programmablauf vom Auftreten bestimmter Kräfte oder Drehmomente beeinflußt werden, dann gibt der Programmierer Grenzwerte an, die zur Laufzeit mittels Kraft- und Drehmomentsensoren überwacht werden. Diese Sensoren sind in einer Meßdose am Greifer untergebracht.

Beispiel: Eine Schachtel wurde ergriffen und soll zur Position POS innerhalb von 5 Sekunden bewegt werden; dabei wird die Position P1 mit einer Geschwindigkeit von 10 cm/sec durchfahren. Tritt während dieser Bewegung eine Kraft größer als 850 Gramm in Z-Richtung auf, so stoppt die Bewegung sofort.

```
MOVE BOX TO POS
     VIA P1 WHERE VELOCITY = 10 * CM/SEC
     WITH DURATION = 5 * SEC
     ON FORCE (ZHAT) > 850 * GM
                DO STOP;
```

Im obigen Beispiel heißt es MOVE BOX und nicht MOVE ARM, obwohl eine Bewegung immer nur vom Arm durchgeführt werden kann. Das AL-System enthält jedoch ein Umweltmodell, bestehend aus den Frames und deren Relationen zueinander. Mit Hilfe von AFFIX/UNIFIX können solche Relationen (Datentyp TRANS) eingerichtet/gelöst werden. Sind zwei Frames über AFFIX miteinander verbunden worden, so wird bei einer Änderung des Wertes des einen Frames auch automatisch der Wert des anderen Frames (vom Laufzeitsystem) in der Art korrigiert, daß die Relation zwischen den Frames erhalten bleibt. Da der Arm auch als ein Frame betrachtet wird, und BOX vorher mit dem Arm durch den Befehl AFFIX BOX TO ARM RIGIDLY verbunden wurde, bedeutet ein Verändern der Position BOX auch ein Bewegen des Armes. Der AFFIX-Befehl wird vor allem für Positionen auf Werkstücken verwendet, die ja auch nach einem Bewegen des Werkstücks relativ zum Werkstück unverändert bleiben, wie z.B. ein Loch in einem Kasten. Für den Programmierer entfallen lästige explizite Angaben und Berechnungen.

Als weitere Fähigkeit von AL sei noch die parallele bzw. quasiparallele Ausführung von Programmabschnitten (Blöcken) erwähnt, die sich durch Semaphore synchronisieren lassen. Dies ist für die Programmierung mehrerer Roboterarme oder die Zusammenarbeit von Roboter, Maschinen und Transporteinrichtungen wichtig.

Implementierung von AL an der Uni Karlsruhe

Da in der Bundesrepublik Deutschland bisher nichts Vergleichbares existiert, wird an der Uni Karlsrune im Zuge eines Technologie-Transfers die höhere Programmiersprache AL implementiert, die an der Stanford University entwikkelt wurde. Das gesamte Programmiersystem AL wurde auf der Rechnerkonfiguration PDP 11/34 - LSI 11/2 vollständig neu implementiert und dient zur Programmierung eines PUMA 500 von Unimation. Aus räumlichen Gründen sind die Rechner über Modems und eine Postleitung verbunden. Der Roboter ist mit einem elektrischen Greifer und einer Kraft- und Drehmomentmeßdose ausgerüstet. Das AL-Programmiersystem besteht aus einem Compiler, der die in AL geschriebenen Programme syntaktisch prüft und in einen AL-Code übersetzt, und einem Interpreter, der den AL-Code ausführt, sowie einer interaktiven Komponente, die ein Definieren der Bewegungspunkte mittels Teach-in erlaubt. Die Implementierung des Compilers erfolgt in PASCAL, wobei der Parser mit Hilfe des PGS (Parser Generating System) der Uni Karlsruhe erstellt wurde. Dazu ist allerdings eine vollständige formale Definition der Syntax von AL in erweiterter Backhus-Naur-Form notwendig. Der vom Compiler erzeugte AL-Code stellt die numerische Verschlüsselung der AL-Befehle dar, daher ist der Interpreter eine abstrakte AL-Kellermaschine. Der AL-Interpreter wird mit Hilfe eines sprachunabhängigen Makrogenerators (eine Erweiterung des PGM von Strachey) und PDP-Assemblers erstellt. Das gesamte System ist daher sehr portabel gehalten und modular aufgebaut, so daß eine Übertragung auf andere Rechner oder Roboter möglich ist. Das System kann zur Programmerstellung in drei verschiedenen Modi gefahren werden:

1) Standard-Modus: Die AL-Programme werden vom Compiler off-line (d.h. ohne Kommunikation mit dem Interpreter bzw. PUMA 500) erstellt und vom Interpreter irgendwann ausgeführt.

2) Test-Modus: Die AL-Programme werden vom Compiler übersetzt und sofort vom Interpreter ausgeführt. Der Anwender kann die Ausführungsgeschwindigkeit bestimmen und das Programm evtl. anhalten. Da er die gerade ausgeführten Befehle auf das Terminal ausgegeben bekommt, kann er entsprechend dem realen Programmablauf Fehler korrigieren und Variablenwerte überschreiben.

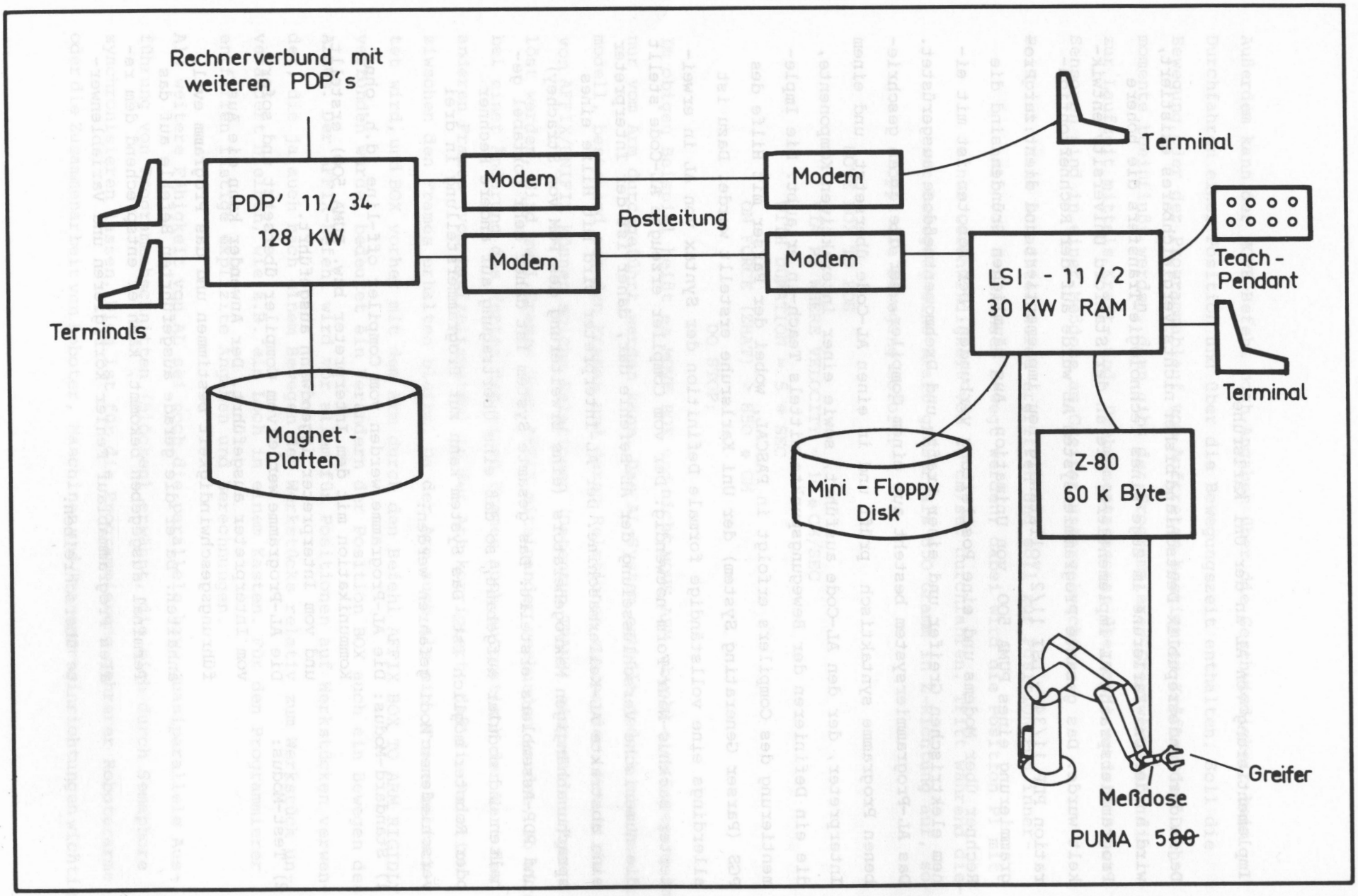

Bild 3 : Roboter-Rechner - Konfiguration an der Universität Karlsruhe

3) Interaktiver Modus: In diesem Modus werden einzelne AL-Statements sofort übersetzt und ausgeführt. Der Anwender kann so ein Programm interaktiv entwickeln und mittels einer Programmiereinheit die Bewegungspunkte definieren. (Diesen Modus realisiert das POINTY-Programmpaket der Stanford University).

Zur Programm-Erstellung ist in jedem Fall der Compiler und die interaktive Komponente notwendig. Im Test-Modus läuft auch der Interpreter, da das AL-Programm sofort ausgeführt wird (Bild 4). Einen Überblick über die Kommunikation zwischen den Softwarekomponenten und zwischen dem Programmiersystem und seiner Umwelt zeigt Bild 5. Dabei wird vorausgesetzt, daß der AL-Programmierer seine AL-Programme mit Hilfe eines Editors erstellt und auf einer Datei abgelegt hat (er kann sie auch direkt an den Compiler übergeben, was nur bei kurzen Programmen sinnvoll ist). Im Standard-Modus übersetzt dann der AL-Compiler das Programm, gibt dabei ein Übersetzungs- und Fehlerprotokoll aus und legt eine Datei mit dem generierten AL-Code an. Diese Datei wird vom Anwender auf die Mini-Floppy-Disk des Steuerrechners transferiert. Der Interpreter liest während der Programmausführung von der Datei den AL-Code und führt ihn aus, d.h. er berechnet die zur Armbewegung notwendigen Steuersignale und schickt sie an die Arm-Steuerung, er liest Sensordaten über die Lage des Arms sowie auftretende Kräfte oder Drehmomente am Greifer ein; er erledigt die vom Anwender geforderte Textein/ausgabe; er reagiert auf externe Unterbrechungen und gibt Signale an externe Geräte aus; er führt mathematische Operationen durch; er steuert den Programmablauf u.a.m.

Zusammenfassend läßt sich sagen, daß mit AL erstmals ein benutzerfreundliches Programmiersystem für Industrieroboter in der Bundesrepublik zu Verfügung steht, das folgende Eigenschaften erfüllt:

- umfangreiche Sprachkonstrukte zur Formulierung komplexer Roboteranwendungen
- umfassende Analyse von Syntaxfehlern sowie Ausgabe von Korrekturhilfen
- selbsttätige Korrektur einfacher Syntax-Fehler
- umfangreiche Testhilfen wie
  - einfaches Verfolgen des (realen) Programmablaufs
  - Anzeige und Korrektur von Variablenwerten
  - Rückgängigmachen des letzten Bewegungsbefehls

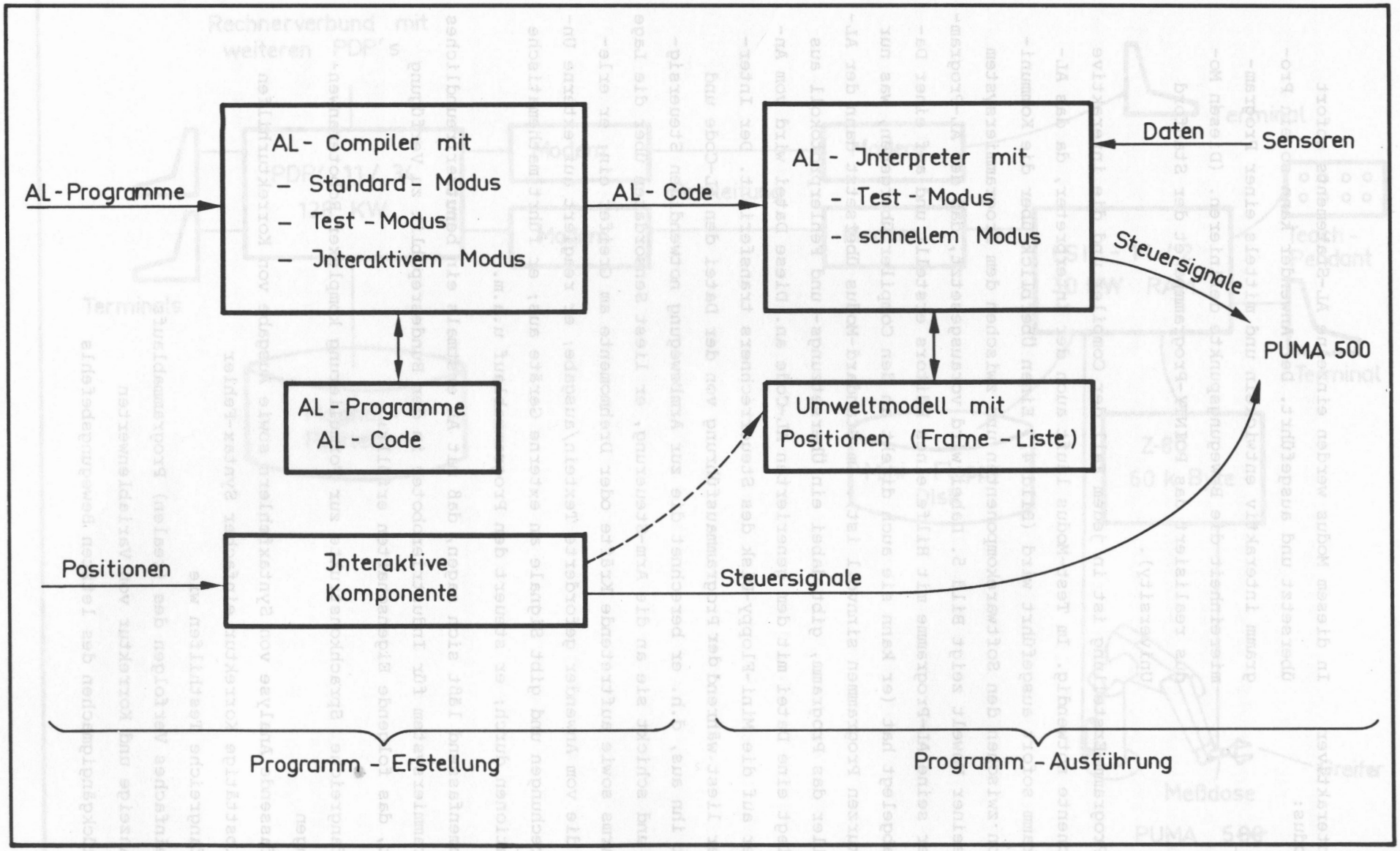

Bild 4 : Software - Komponenten der AL - Implementierung

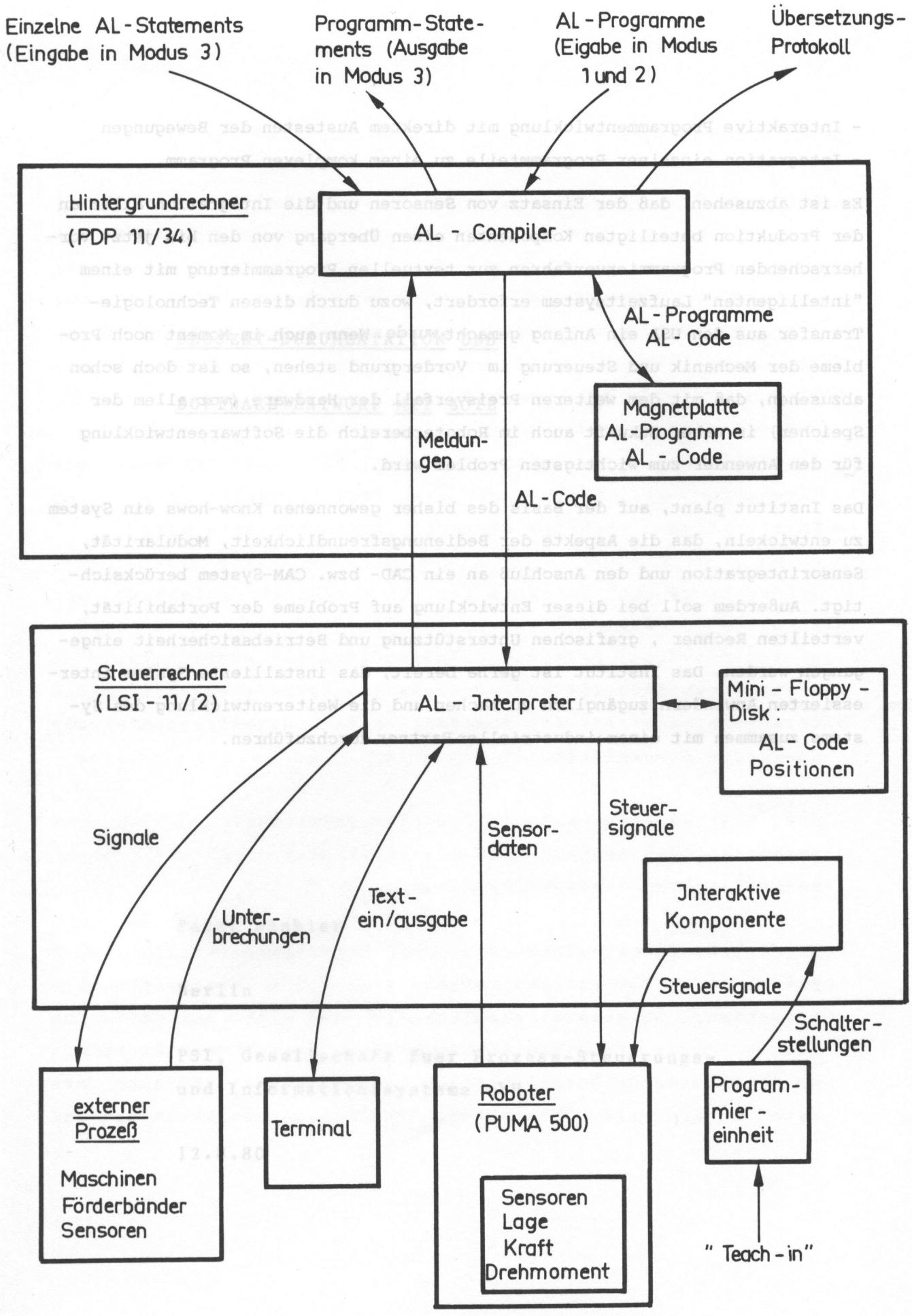

Bild 5: Übersicht über das AL-Programmiersystem

- Interaktive Programmentwicklung mit direktem Austesten der Bewegungen
- Integration einzelner Programmteile zu einem komplexen Programm.

Es ist abzusehen, daß der Einsatz von Sensoren und die Integration aller an der Produktion beteiligten Komponenten einen Übergang von den bis jetzt vorherrschenden Programmierverfahren zur textuellen Programmierung mit einem "intelligenten" Laufzeitsystem erfordert, wozu durch diesen Technologie-Transfer aus den USA ein Anfang gemacht wurde. Wenn auch im Moment noch Probleme der Mechanik und Steuerung im Vordergrund stehen, so ist doch schon abzusehen, daß mit dem weiteren Preisverfall der Hardware (vor allem der Speicher) in naher Zukunft auch im Roboterbereich die Softwareentwicklung für den Anwender zum wichtigsten Problem wird.

Das Institut plant, auf der Basis des bisher gewonnenen Know-hows ein System zu entwickeln, das die Aspekte der Bedienungsfreundlichkeit, Modularität, Sensorintegration und den Anschluß an ein CAD- bzw. CAM-System berücksichtigt. Außerdem soll bei dieser Entwicklung auf Probleme der Portabilität, verteilten Rechner , grafischen Unterstützung und Betriebssicherheit eingegangen werden. Das Institut ist gerne bereit, das installierte System interessierten Anwendern zugänglich zu machen und die Weiterentwicklung des Systems zusammen mit einem industriellen Partner durchzuführen.