

**P**rocess and  
**E**xperiment  
**A**utomation  
**R**ealtime  
**L**anguage

# PEARL

**P**rogrammieren von  
**E**chtzeit-  
**A**nwendungen  
**R**elativ  
**L**eicht

Das außerordentliche Anwachsen der Software-Kosten fordert den Übergang von der veralteten Assemblerprogrammierung zur strukturierten Programmierung in einer höheren Echtzeit-Programmiersprache. **PEARL** ist weltweit die einzige anwendungsorientierte höhere Echtzeit-Programmiersprache. Unabhängige Probleme können als eigenständige Prozesse (Tasks) programmiert und parallel exekutiert werden – eine wesentliche Erleichterung bei der Realisierung automatisierungstechnischer Probleme.

Die Anfänge von **PEARL** liegen in den 70er Jahren. Ziel der vom BMFT geförderten Entwicklung war die Definition einer Sprache, die die wichtigsten Elemente der gängigen Hochsprachen mit einem klaren Realzeit- und Tasking-Konzept vereinigt. **UH-PEARL** ist eine Implementierung der Sprache auf Mikroprozessorsystemen, deren Entwicklung Anfang der 80er Jahre an der Universität Hannover unter Leitung von Prof. Dr.-Ing. W. Gerth begann.

**PEARL** ist eine leicht erlernbare Programmiersprache insbesondere zur Lösung echtzeitorientierter Probleme. Sie ist, anders als zum Beispiel Prozess-FORTRAN, eine Sprache aus einem Guß, die direkt integrierte Sprachelemente für Prozess-E/A und zeitliche Aufgabeneinplanung besitzt. Damit ist ein hohes Maß an Portabilität gegeben.

**PEARL** ist eine blockorientierte, strukturierte Sprache. Sie ist universell auch zur Lösung komplexer, algorithmischer Probleme geeignet. **PEARL** stellt Sprachelemente zur Interrupt-Behandlung und zur synchronisierten Kommunikation zwischen Tasks zur Verfügung.

Ein ausgearbeitetes Multitasking-Konzept, die Unterstützung aller gängigen algorithmischen Kontrollstrukturen sowie leicht anwendbare Echtzeit-Sprachelemente sind wesentliche Merkmale des **UH-PEARL**. Bei der Entwicklung der Sprache wurde besonderer Wert auf gute Selbstdokumentation und einfache, klare Formulierung des Echtzeit- und Multitasking-Verhaltens gelegt. Anders als z.B. Ada ist **PEARL** von anwendungsorientierten Entwicklern in kurzer Zeit erlern- und wirkungsvoll einsetzbar.

## Warum PEARL

**modernes  
Konzept**

**leicht erlernbar**

**universell  
einsetzbar**

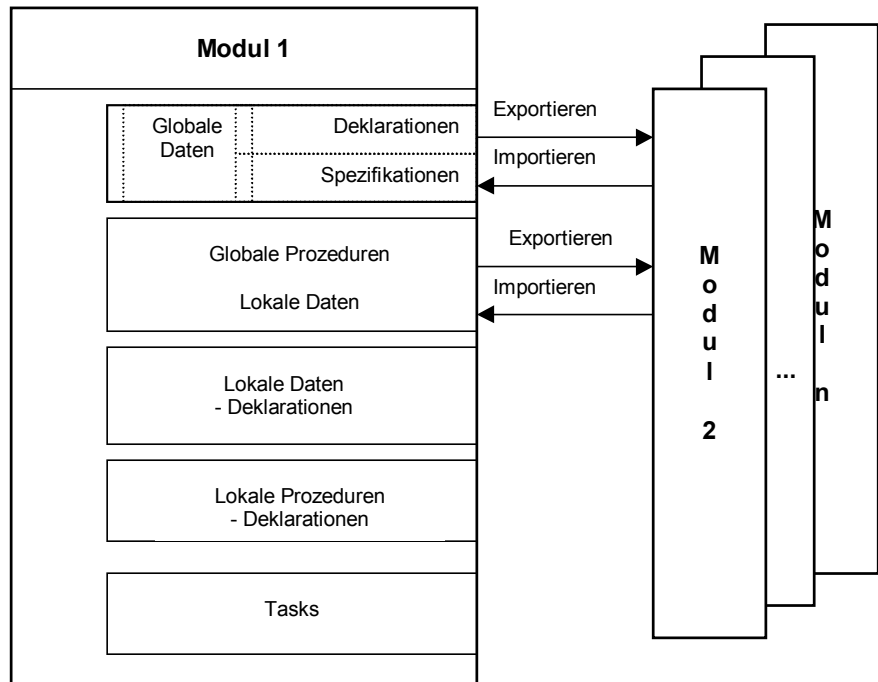
**Sprach-  
eigenschaften**

---

---

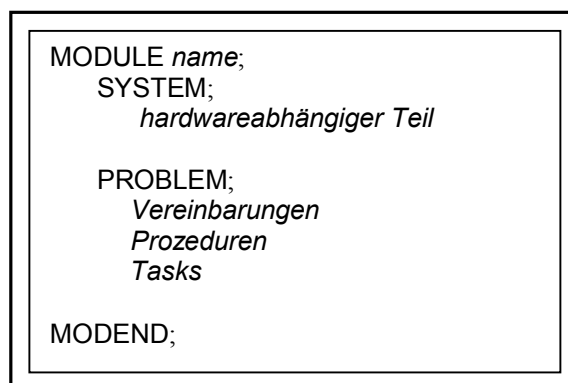
## Modularität

Die Voraussetzung für sichere und effiziente Programmentwicklung und Wartung bei größeren Projekten schafft der modulare Aufbau von **PEARL**-Programmen.



## Portabilität

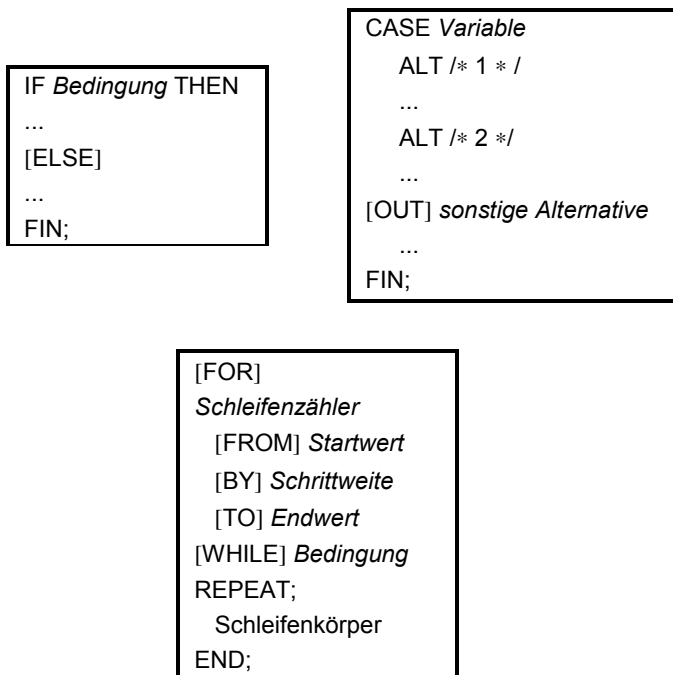
Die Unterteilung eines Moduls in einen hardwareabhängigen **SYSTEM**-Teil, in dem die Verbindung zwischen den **PEARL**-Datenstationen und der Umwelt definiert wird, und dem hardwareunabhängigen **PROBLEM**-Teil erleichtert Portierungen von **PEARL**-Programmen in eine andere Hardware-Umgebung.



---

Die Aufgliederung eines Moduls in hierarchische Blöcke mit Kapselung lokaler Daten bietet einen problemangepaßten Programmaufbau. Quasi-parallele Abarbeitung der Tasks und wiedereintrittsfeste wie auch rekursive Prozeduren ermöglichen die einfache Aufteilung eines Problems in unabhängige Prozesse.

**PEARL** unterstützt alle bei modernen Sprachen üblichen algorithmischen Elemente zur Kontrolle des Programmflusses:



Neben den auch in anderen Sprachen üblichen Grunddatentypen FIXED, FLOAT, CHAR stellt **PEARL** außerdem CLOCK (Uhrzeit), DURATION (Zeitdauer), SEMA (Synchronisationsvariable) und BIT (Bitkette) zur Verfügung. Neue Datentypen können durch problem-spezifisches Zusammenfassen von Elementen verschiedener Grunddatentypen zu Verbunden (STRUCT) und durch eigene Typ-Vereinbarungen (TYPE) definiert werden. Zusammen mit Zeiger-variablen (REF) erreicht man eine fein abgestufte Modularisierung der Programme.

**PEARL** ist seit 1981 in DIN 66 253 Teil 1 Basic PEARL und seit 1982 in DIN 66253 Teil 2 Full PEARL genormt und hat sich bereits in über 200 Groß- und vielen hundert Kleinprojekten bewährt.

Mit der Normung von PEARL-90 in DIN 66253-2 wurde das Sprachkonzept den aktuellen Anforderungen entsprechend erweitert. Die aktuelle Weiterentwicklung von **PEARL** versucht, objekt-orientierte Programmierverfahren mit den Sicherheits- und Effizienz-anforderungen der Echtzeitprogrammierung zu vereinbaren.

## Blockstruktur

## Kontrollstrukturen

## Datentypen

## Normung

---

## Echtzeit- anweisungen

Die einfache zeitliche Organisation des Programmablaufs

```
AFTER 10 SEC
ALL 4 SEC
UNTIL 17:00:00
ACTIVATE Regler PRIO 6;
```

Zyklische Durchführung  
der Task Regler bis zu  
einer festen Uhrzeit

sowie die integrierte Interruptverarbeitung

```
WHEN Feuer ACTIVATE Loesch;
```

zur Einplanung der Task Loesch auf den Interrupt Feuer erfolgt  
durch quasi selbstdokumentierende Anweisungen.

## Ein-/Ausgabe Anweisungen

Ein-/Ausgaben erfolgen über stets identische **PEARL**-Daten-  
stationen. Rechnerspezifische Kontroll- und Konvertierungsfunk-  
tionen werden über Vereinbarungen nur im SYSTEM-Teil be-  
stimmt. Insbesondere Prozess-E/A

```
SEND Aus TO Motor;
```

```
TAKE belegt FROM Lichtschranke;
```

und dateiorientierte, alphanumerische E/A

```
PUT Temperatur TO Protokolldatei;
```

```
GET Sollhoehe FROM Bedienkonsole;
```

können so leicht an unterschiedliche Gerätekonfigurationen ange-  
paßt werden.

## ROM-Code

Der **PEARL**-Compiler erzeugt optimal ROM-fähigen Code und  
erlaubt so die Programmierung abgeschlossener Systeme ohne  
Massenspeicher. Beim Start des Systems werden nur die ggf. mit  
Initialwerten besetzten Variablen im RAM angelegt; der Pro-  
grammcode wird aus dem ROM heraus exekutiert.

## Verfügbarkeit

IEP unterstützt den Einsatz von UH-**PEARL** auf allen Rechnern  
unter dem Betriebssystem RTOS-UH. Zur Zeit sind dies

- alle Prozessoren der Motorola-MC68000-Familie (MC68000 –  
MC68060, Controller der MC683xxx-Reihe)
- Prozessoren der PowerPC-Familie (MPC603, MPC604,  
MPC750, Controller der MPC5xx, MPC8xx und MPC82xx-  
Reihe)

Der Leistungsumfang der Systeme reicht von kleinen embedded-  
control-Sonderentwicklungen bis hin zu vernetzten Mehrprozes-  
sorsystemen auf der Basis von Standardbussen. Der UH-**PEARL**-  
Compiler kann auch als Cross-Compiler eingesetzt werden. Ver-  
sionen für z.B. Microsoft-Windows seit Windows 95, VAX-VMS,  
UNIX, DOS etc. sind erhältlich; der erzeugte Code kann auf jedem  
Rechner unter RTOS-UH ausgeführt werden.