

Process and
Experiment
Automation
Realtime
Language

PEARL

Programmieren von
Echtzeit-
Anwendungen
Relativ
Leicht

The extraordinary increasing of software costs especially in realtime applications demands the transition from outdated assembler programming to a higher level structured programming language. **PEARL** is the only application orientated higher realtime programming language world-wide. Independent problems can be programmed as independent processes (tasks) and executed in parallel – a substantial improvement within the field of control engineering.

PEARL was born in the 1970's. A goal of the development promoted by the German Ministry of Research and Technology was the agreement on a language, which combines the most important elements of the common high-level languages with a concise concept of realtime and tasking. **UH-PEARL** is an implementation of this language for microprocessor systems, the development started in the beginning of the 1980's at the University of Hanover under the leadership of Prof. Dr. Ing. W. Gerth.

PEARL is an easy-to-learn programming language suited in particular to solve realtime-oriented problems. It is, to differentiate from e.g. process FORTRAN, a monolithic language that directly integrates process I/O and time-oriented task scheduling. Thus, a high measure of portability is given.

PEARL is a block oriented, structured language. It is universally suitable also to solve complex, algorithmic problems. Apart from all common programming language elements, **PEARL** integrates interrupt handling and synchronization objects.

A well formulated concept of multitasking, the support of all usual algorithmic control structures as well as concise realtime control statements are the special features of **PEARL**. During the language design, special attention was paid to support the writing of programs, that are easy to read and, therefore, easy to maintain, without restricting the developer or detracting him from his problem domain. Differently than e.g. Ada, **PEARL** is orientated towards the applications engineer and gives an easy start without a steep learning curve.

Why PEARL

A modern Concept

Easy to learn

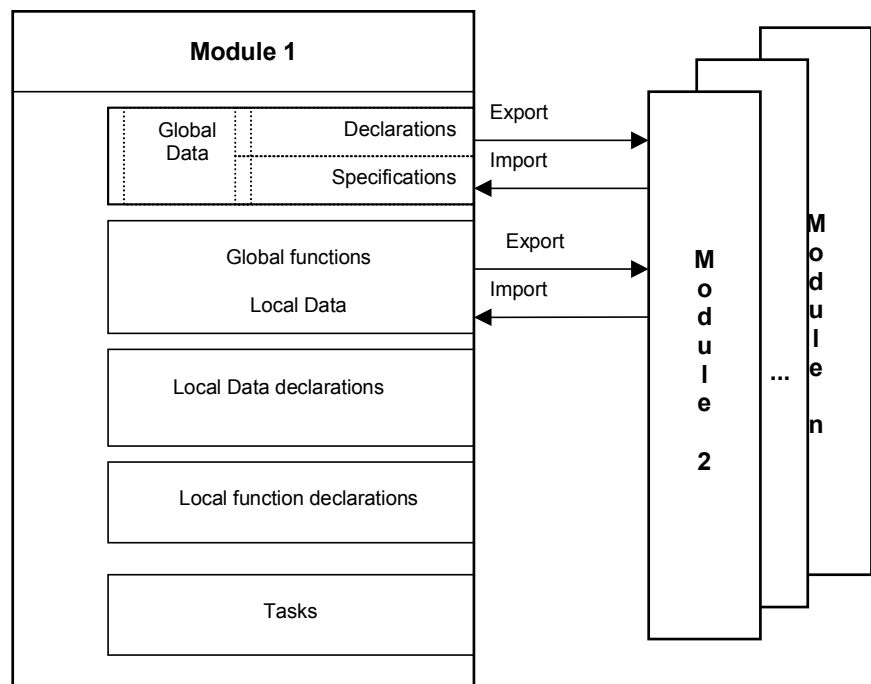
Universally applicable

Language characteristics

Modularity

The modular structure of **PEARL** programs lays ground for safe and efficient program development and easy maintenance within larger projects.

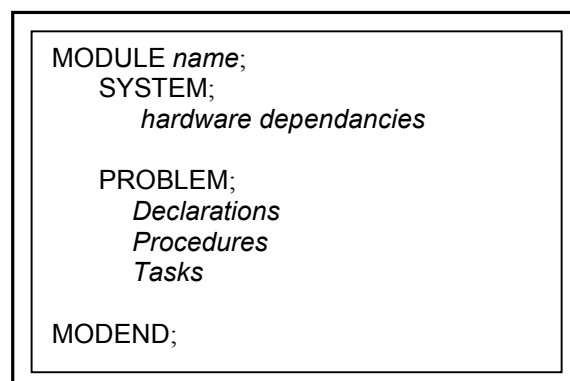
A Module is a self contained compilation unit, but not necessarily a self contained execution unit. The connection between modules is made by global declarations and the corresponding specification of global symbols. Inter-module relations can be satisfied either by an additional linker or by load-time linkage.



Portability

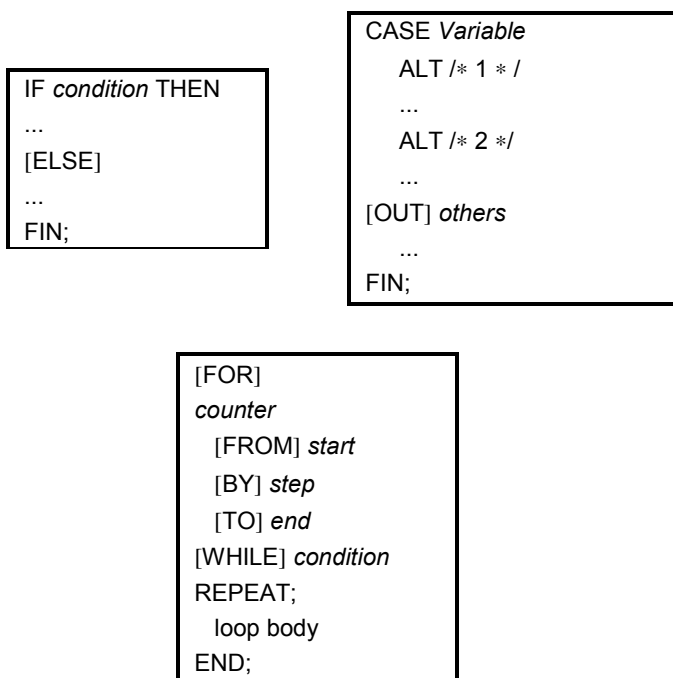
One **PEARL**-Module is separated in two parts, each of which may be omitted. A module starts with a **SYSTEM**-part, defining the system resources used in a system dependant manner. This part contains no code.

The following **PROBLEM**-part is system independant and may use only system services defined in either its own or another module's **SYSTEM**-part.



The breakdown of a module into hierarchical blocks with local data allows for a program structure that mirrors the structure of the problem. Quasi parallel processing of tasks serve the ease of separating a problem in independent and simpler pieces of code. Procedures, which are reentrant and allow for recursion, provide for the hassle-free realisation of problem specific code libraries.

PEARL supports all flow control structures common to modern programming languages:



Apart from the regular simple data types FIXED, FLOAT and CHAR, also common to other languages, **PEARL** provides the additional data types CLOCK (time), DURATION (length of time), SEMA (synchronisation variable) and BIT (bit string) to allow for strong typing in the problem domain. New data types can be defined by problem-specific combination of elements of different basic data types into groups (STRUCT) and by own type declarations (TYPE).

PEARL is standardized since 1981 in DIN 66,253, part 1, Basic PEARL, and since 1982 in DIN 66253, part 2, Full PEARL. PEARL is already used in over 200 large and many hundred small projects.

In 1998, with the standardization of PEARL-90 in DIN 66253-2, with caution the concept of the language was adapted to current requirements. The current advancement of **PEARL** tries to agree upon object-oriented programming procedures in combination with the safety and efficiency requirements of realtime programming.

Block structure

Control structures

Data types

Standardization

Realtime statements

The simple time scheduling of program flow

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

Scheduling the cyclic execution of the task control in a given time-frame

as well as the integrated interrupt scheduling

```
WHEN fire ACTIVATE douse;
```

to schedule task douse to execute when the interrupt fire is triggered allow for comprehensive self-documenting instructions.

I/O statements

In **PEARL**, input/output statements use so-called DATIONS. System-specific properties of dations are declared in a module's SYSTEM part; in the PROBLEM part, these dations are used in a portable manner. So, I/O of process values, e.g.

```
SEND Off TO engine;
```

```
TAKE is_engaged FROM clutch_switch;
```

and file-oriented, alphanumeric I/O

```
PUT temperature TO log_file;
```

```
GET target_height FROM console;
```

can be ported easily to different device configurations.

ROM code

The **PEARL** compiler can generate ROM-able code and supports targets without mass storage. At the start of the operating system, during the phase of self-configuration, programs in ROM are recognized, their RAM-Areas get initialised and their code ist executed directly out of the ROM.

Availability

IEP supports the deployment of **UH-PEARL** on all computers under the operating system RTOS-UH, based on e.g.

- the MC68xxx-family (MC68000 –MC68060, MC683xxx)
- the PowerPC family (MPC60x, MPC750, MPC5xx, MPC8xx,...)

The capabilities of these systems cover small embedded controls as well as high-powered multiprocessor-systems based on e.g. commercial off-the-shelf VMEbus-boards.

The **UH-PEARL** compiler is available either generic or as cross-compiler, runnable under all versions of the Microsoft Windows operating system since Windows '95.