PASCAL? ADA?? PEARL!!!

Process and Experiment Automation Realtime Language in Industrial and University's Environment PEARL on HP3000/HP1000 Networks

Klaus Rebensburg
Technical University of Berlin (West Germany)

INTRODUCTION

Steadily increasing software costs and the inherent risks computer automation projects made it necessary to replace the obsolete technology of assembler programming by structured programming in a high-order real-time language. At present, PEARL is the world's most powerful and advanced high-order real-time language used for industrial process automation.

PEARL has been developed for application engineers. In comparison to system implementation languages like Concurrent PASCAL, MODULA or ADA, PEARL is the only real-time language which the user can learn quickly and apply readily. In contrast to (Process-) FORTRAN, PEARL is a homogeneous language which e.g. has built-in language elements for process input/output and for task-scheduling. Thus PEARL offers a higher degree of portability than any other language for process automation.

At the Technical University of Berlin PEARL is implemented on a HP3000 Series II and HP1000 F in autonomous versions and in the university's process-control network as cross-version between HP3000→HP1000 computers.

PASCAL?

PASCAL offers good tools for structured programming. For a lot of applications the user needs tasking facilities and process-input/output. Built-in language elements are the best solution to program such problems. PASCAL does not have these scheduling and I/O elements.

ADA??

ADA is a high order programming language for so-called "embedded systems." ADA was developed in order and under control of the Department of Defense (DOD) and is supposed to become a widely spread programming language. ADA offers good tools for structured programming. Especially the attempt to define the compiler environment is a good approach for cost-effective software-production.

Multiprogramming facilities are provided with a minimal set of basic operations. No attempt is made to define special features covering the large range of input/output applications. The language facilities are designed in a way that the user has to provide I/O packages to define special process-I/O features.

That means there is no standardization for process-I/O in ADA. Until now there is no official ADA-compiler system running. Software producers have to wait — how long? There are some problems to make ADA learnable for application engineers, especially the so-called rendesvouz-concept might be difficult to map it on real technical processes. There are some other problems to develop ADA programs on minicomputers as ADA is designed for cross-software development on big computer systems.

How many years must a HP-user wait for ADA3000 or ADA1000?? Remember — PASCAL was defined in 1971, now in the 1980s we will look forward to use the official PASCAL on HP3000. ADA was defined 1979.

PEARL !!!

- Algorithmic language elements
- Abstract data types
- Modular program structure
- Real-time language elements
- Description of the hardware-configuration / Input/output language elements
- Who uses PEARL
- Standardization of PEARL
- Availability of PEARL for Hewlett-Packard Computers

What Are the Characteristics of PEARL?

PEARL offers all language characteristics which the user needs in order to solve his industrial automation problems. Some software houses use PEARL as implementation language for all kinds of greater software-problems, like database-management, big software-packets.

Algorithmic Language Elements

The language elements for the formulation of algorithms and procedures correspond to the state of the art of modern programming languages (e.g., PASCAL).

Algorithmic elements can be written as declarations of procedures, functions, and user-defined operators (!) and tasks (!). (The last two features are e.g. not provided by PASCAL.)

Tasks are elements for parallel execution.

List of PEARL statements:

```
variable := expression;              ( assignation )
GOTO   identifier;
CALL   proc_identifier;              ( procedure call )
RETURN (expression);                 ( return from procedure,function )
INDUCE signal_identifier;            ( raise an exception )

ON signal_identifier: statement;     ( reaction on exception )

BEGIN  declarations                  ( block )
       statements
END;

IF condition                         ( if-then-else construct )
   THEN statements
   ELSE statements
FIN;

CASE expression                      ( case construct )
  ALT statements
  ...
  OUT statements
FIN;

FOR variable                         ( for-while-loop construct )
  FROM expr BY expr TO expr
  WHILE condition
    REPEAT declaration_list
           statements
    END;
```

Abstract Data Types

The modern concept of data types in PEARL enables the user to define problem oriented, composite data types and new operators. These abstract data types permit a great number of checks at compile time and contribute to a refined modular program structure (strong typing).

```
basic types: FIXED                  ( fixed )
             FLOAT                  ( floating point )
             BIT                    ( bit )
             CHAR                   ( character )

             REF                    ( pointer )
             CLOCK, DURATION        ( real-time !! )
             SEMA,  BOLT            ( semaphores, bolts !! )

structures:  array  (dynamic)
             STRUCT
             bit chain
             DATION                 ( standard and user-defined
                                      peripherals )

             (structures can be nested)


allocation of variables:
             at address determined by compiler
             RESIDENT attribute indicates fast access
```

```
access to variables:
            by name or reference

standard operators:
            many standard operators are defined in the langage,
            e.g., ABS, LWB, TOFIXED, TOFLOAT, TOBIT, TOCHAR,
            and CSHIFT.
```

Modular Program Structure

A PEARL program is composed of separately compilable modules with exactly defined interfaces. This structure greatly facilitates communication between the members of a project team and supports the modular composition of complex program-systems.

Example:

```
MODULE (reportwriter); /* written by programmer A */
  SYSTEM;
    /* contains description of configuration */
  PROBLEM;
    /* specification of imported objects ;
       declaration of tasks,procedures,operators,data,types,dations */
    ...;
MODEND; /* end of module reportwriter */

MODULE ( brewery_control); /* written by programmer B */
  PROBLEM;
     ...
MODEND; /* end of module brewery_control */
```

Real-time Language Elements / Multiprogramming Facilities of PEARL

For programming task scheduling, PEARL contains real-time language elements, which can be learnt and applied easily.

Example:

```
AFTER 5 SEC ALL 7 SEC DURING 106 MIN ACTIVATE relay PRIORITY 5;
(5 seconds after the execution of this statement the computing
 process 'relay' is activated with priority 5 every 7 seconds
 for a total period of 106 minutes)


extended_time_specification ACTIVATE task; ( scheduling )
TERMINATE task;
SUSPEND   task;
time_spec CONTINUE task;
time_spec RESUME   task;
PREVENT task;

operations on semaphores:          ( synchronization, communication )
  REQUEST, RELEASE
operations on bolt variables:      ( multiple reader-writer problems )
  RESERVE, FREE, ENTER, LEAVE
operations on interrupts:          ( interrupt-handling )
  DISABLE, ENABLE, TRIGGER

  WHEN interrupt_identifier task_control_statement;

operations on signals:             ( exceptions )
  ON signal: statement;            ( react upon exception )
  INDUCE signal;                   ( raise an exception )

Example: At 16:00:30 RESUME task; ( delay )

         WHEN interrupt_id AFTER 10 SEC EVERY 20 MIN UNTIL 15:20:00
         ACTIVATE task_id;
```

Description of the Hardware-configuration / Input/output Language Elements

In the System-Division of PEARL the hardware configuration, especially process peripherals, can be described independently of the special hardware realisation by user defined identifiers. This capability greatly enhances documentation value and portability of PEARL programs.

Example:

```
MODULE (demo); /* this module contains a PEARL - program */

SYSTEM;

  display: stdio(1); /* display is the identifier used in the
                        program for standard-I/O. stdio(1) is a
                        system defined name for the HP3000
                        implementation                          */

  disk   : disc(3);  /* the user disk */

  engine : prog_cont;/* engine is the identifier used in the
                        program for 16 Bit I/O. The HP3000 knows
                        it as programmable controller interface   */

PROBLEM;

  /* we start with the specification of the peripherals */

  SPECIFY display   DATION INOUT ALPHIC     CONTROL(ALL),
          disk      DATION IN    ALL DIM(,) CONTROL(ALL),
          engine    DATION OUT   BASIC;

  /* now we use these peripherals */
  STARTtest: TASK;

  DECLARE on        INV BIT(16) INIT ('0000000000000111'),
          off       INV BIT(16) INIT ('0000000000000001');

  /* let's switch the engine */

  SEND on TO engine;
  AFTER 2 MIN RESUME;    /* continue 2 minutes later */
  SEND off TO engine;
  PUT `we have stopped the engine' TO display by SKIP,A(30),SKIP;

  END; /* end of task STARTtest */
  ...
MODEND; /* end of module */
```

Another example for PEARL I/O:

```
TAKE pressure FROM pressure_sensor;

SEND open TO valve;     /* output of the value `open' to device
                           `valve' */
```

Input and output features: Data-stations (DATIONs), generalizing real or virtual peripherals or I/O channels. Interfaces, mapping data-stations with different properties onto each other to offer the possibility to define formatting routines (objects of type CONTROL).

Most of these PEARL systems don't use Hewlett-Packard computers (?). PEARL compilers are offered by many (mostly European) computer manufacturers and software houses.

An object of type DATION represents in general a set of one to four channels:

Data	channel (transfer values of PEARL objects)
Control	channel (transfers values of type CONTROL)
Interrupt	channel (signals events of type INTERRUPT)
Signal	channel (signals events of type SIGNAL)

Who Uses PEARL?

PEARL is already widely used (with applications mostly in W-Germ Process computer projects in many different areas have been sucessfully programmed with PEARL.

The following table is a survey of PEARL activities (1980):

Field of Application	Number of PEARL Systems
Metal Processing, Rolling Mills	41
Power Distribution	36
Power Generation	5
Raw Materials, Chemical Industry	21
Water Supply	14
Other Industrial Applications	21
Mail Order Houses, Warehouses	12
Television, Transport, Aerospece Applications	21
Development and Education	32
	204

Following computer systems are available with PEARL compilers: AEG 80-20, AEG 80-60, Data General NOVA, DP1000/1500 by BBC PDP 11 family, Krupp-Atlas EPR 1100/1300/1500, Hewlett-Packard HP1000 and HP3000, Interdata 7/32, INTEL 8086, LSI 11, Micronova MUC161, MUDAS 432, MULBY 3, NORD 10 by Norsk Data, Digital Equipment PDP11 family, PDP 11/34, Siemens 300,310,404/3, VAX 11/780, Z80, MOTOROLA MC68000.

Standardization of PEARL

The standardization of PEARL ensures its uniformity. The draft standard DIN 66253 part 1 "BASIC PEARL" has been available since 1978. Draft standard DIN 66253 part 2 "FULL PEARL" followed in August 1980. Parallel to these activities, PEARL has been submitted to ISO for international standardization (TC97/SC5/WG1). For these purposes PEARL was described completely with petri-nets and attributed grammars (formal syntax and semantics).

Availability of PEARL for HP Computers

The Technical University of Berlin uses PEARL in three different ways. The university's real-time process-control computer network consists of 1 HP3000 series II 512 KByte, 16 HP1000 computers (M,E,F) and 30 microcomputer systems. The central computer HP3000 is used for program-development, documentation, cross-software, statistics, graphics, education, whereas the decentral HP1000 systems are used for process-control applications.

a) PEARL3000 is the autonomous PEARL Programming System on HP3000. It is used for program development and training courses. Typical courses are visited by 18 persons each, 3-4 teachers 7 terminals, 5 days mixed theory, practical work, discussions. Participants are EDP leaders, engineers, programmers from industry and university.

The Technical University Berlin is responsible for design and execution of the official PEARL courses, offered by the VDI (Verein der Ingenieure) Germany and by the PEARL Association Duesseldorf.

In connection to the compiler system a PEARL Testsystem on language level is available. It simulates, e.g., tasking, checks for deadlocks, I/O. PEARL3000 is running under MPE4 on HP3000 Series II/III computers.

b) PEARL1000 is the autonomous PEARL Programming System on HP1000. It is used for program development, training courses, real-time and process-control applications. Most of the PEARL1000 implementation was developped by the Technical University of Berlin. The implementation was sponsored by the Ministry of Research and Technology of (West-) Germany. PEARL1000 is running under RTE4B with HP1000 F Computers (256 KBytes)

c) PEARL3000/1000 is the Cross PEARL System for HP3000/HP1000 networks. Compilation, interface-check, relocation is done on HP3000 with the obvious advantage that during the coding phase many programmers can work simultaneously.

The cross-version supports small HP1000 configurations, core-resident systems which are autonomous in process-control and loosely coupled with V.25 lines to a HP3000 multiplexor for transfer of relocated code to HP1000 and process-data to HP3000. PEARL3000/1000 supports HP1000 computers under RTEM3 operating systems.

The HP3000 has proven to be a good development computer system for all kinds of other computers. We not only use it for HP1000 computers but also for more than 14 different types of microprocessors.

In all three cases the same PEARL compiler is used. Code generation in each case is adapted to the target-computer. Of course runtime routines and operating system kernel are tailored to the different HP standard operating systems.

No change of MPE or RTE operation system was necessary for the implementation. (Cross-version has some minor changes of RTEM3 operating system.)

Use and handling of the PEARL Compilation Systems is easy and comparable to FORTRAN3000 or other HP subsystems.

REFERENCES

1. DIN 66253 Part 1 "BASIC PEARL" (draft standard, language: English). 1978 Beuth Verlag GmbH Berlin 30, W-Germany.
2. DIN 66253 Part 2 "FULL PEARL" (draft standard, language: English). 1980 Beuth Verlag GmbH Berlin 30, W-Germany.
3. Werum, Wulf: Windauer, Hans; PEARL Process and Experiment Automation Realtime Language. 1978 Vieweg Verlag Braunschweig; Book, which describes HP3000/HP1000 PEARL language. Language: English and German
4. Kappatsch, A.; Mittendorf, H.; Rieder, P. PEARL Systematic

description for the application engineer. R. Oldenbourg Verlag Muenchen Wien 1979. Language: German

5. Kappatsch, A., PEARL Survey of language features. Kernforschungszentrum Karlsruhe KfK-PDV 141 August 1977. Language: English
6. Martin, T.; PEARL AT THE AGE OF THREE. 4th International Conference on Software-Engeneering, September 17-19, 1979 Munich, Germany IEEE No.79 CH1479-5G. Language: English.
7. Martin, T.; PEARL AT THE AGE OF FIVE. Updated Version, published in *Computers in Industry,* Vol. 3, Number 2, 1981. Language: English.
8. Hommel, G.; Experience with PEARL in Industrial Applications. VDE-Congress, Berlin 6.-9.10. 1980. Language: German.
9. Windauer, H.; Development and Implementation of Portable Compilers for Realtime Languages. Proceedings of Real-Time Data '79 Berlin Oct. 1979. Language: English.
10. Martin,T.; Realtime Programming Language PEARL Concepts and Characteristics. Proceeding 2nd Computer Software and Applications Conf., Chicago, 1978, pp 301-306 IEEE Cat.No.78CH 1338-3C.
11. Brinkkoetter, H., Groessler,J., Nagel, K., Nebel, H., Kneuer,E., Rebensburg, K.; PEARL on Hewlett-Packard Computers Implementation and Demonstration. 27.4.1981 Berlin. Language: German.
12. Rebensburg, K.; Real-time Computing with the Process-Control Computer Network of the Technical University Berlin. Language: English.
13. Brinkkoetter, H., Nagel, K., Nebel, H., Rebensburg, K.; Systematic Programming with PEARL. PEARL Training Course Handbook, VDI/PEARL Association, 1981 Berlin. Language: German.
14. PEARL Association; PEARL-RUNDSCHAU. Official bimonthly publication by the PEARL Association Verein. Contains contributions of PEARL applications, software-houses, educational aspects, scientific applications, PEARL News etc. Language: German.

All information about PEARL can be obtained from the author and from:

PEARL Association
Graf-Recke-Strasse 84
4000 Duesseldorf 1

P.S. PEARL information is also available in Spanish, Portuguese, Chinese, French and Serbocroatic languages.