Technische Universität Berlin

PROZESSRECHNERVERBUND - ZENTRALE
Leiter: Dipl.-Ing. K. Rebensburg

Telefon:
(030)314 5154/61
Sekr.: HE 3
Einsteinufer 19
D-1000 Berlin 10
GERMANY

INTRODUCTION OF P E A R L

(Process and Experiment Automation Realtime Language)

ON HEWLETT-PACKARD COMPUTER SYSTEMS

Dipl.-Ing. Jörg Grössler

Contents:

Dieser Bericht veröffentlicht Ergebnisse aus einem mit Mitteln des Bundesministers für Fprschung und Technologie (Kennzeichen DV 5.505) geförderten Forschungsvorhabens des Projektes "Prozeßlenkung mit DV-Anlagen (PDV)" im Rahmen des 3. DV-Programmes der Bundesregierung.

Die Verantwortung für den Inhalt liegt ausschließlich bei den Autoren bzw. den geförderten Unternehmen.

1. Programming Requirements for Process Control

General Requirements

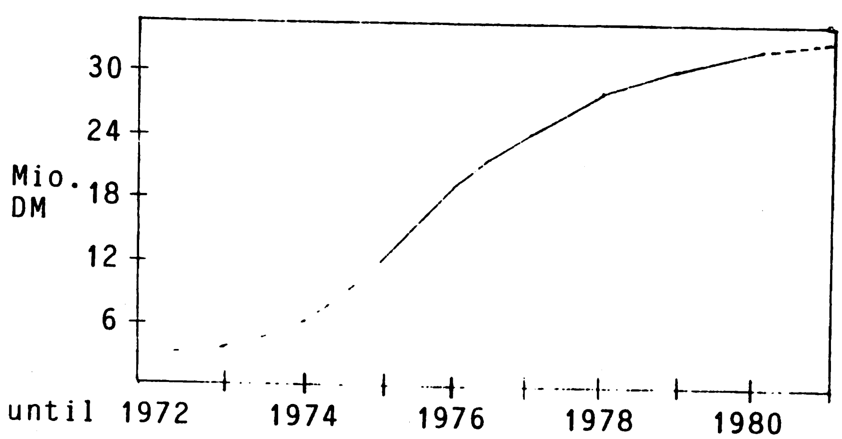
- o Process control programming should not require too much knowledge of the computer internals
- o The programming language should allow economical writing and testing of programs
- o Source code should be easy to read
- o The programming language should support portability
- o Open ended design of the computer control system should be supported through a module oriented program structure
- o The programming language should assist the construction of correct programs and the development of reliable control systems
- o Object code should run sufficiently efficient

Special Requirements concerning Realtime Application

- o Means to describe parallel activities of a program
- o Possibility to describe configuration and capabilities of the system resources to be used in the program
- o Special data types and operators for realtime process control

2. History of PEARL Development

1969	PEARL workshop is founded composed from the start of users, process computer manufacturers, software suppliers, research institutes, and representators of the German Society of Engeneers (VDI/VDE).
1972	Project PDV (directed by a project staff at the Nuclear Research Center, Karlsruhe, on behalf of the Federal Ministry of Research and Technology) takes PEARL under its wings.
1973	Preliminary definition of PEARL is set up.
1976	Final PEARL definition is completed.
1978	DIN Draft Standard of Basic PEARL is published.
1978	Basic PEARL is been submitted to the international working group TC97/SC5/WG1 as candidate for ISO Standard.
1979	DIN Draft Standard of Full PEARL will be published until the end of the year.



PEARL is sponsored by the project PDV on behalf of the Federal Ministry of Research and Technology

Status of PEARL Implementations:

Firm	Process Computer Family
AEG-Telefunken	80-20/40/60
BBC	DP 1000/1500
Siemens	330/340
Dietz	Mincal 621
Krupp-Atlas	EPR 1100/1300/1500
MBP	HP 3000
MBP	Siemens 404/3
announced:	
DIGITAL	DEC PDP 11-Family
Werum	Norsk Data Nord 10S
Werum	MODCOMP
TU-Berlin (Werum)	HP 21MX
announced for microcomputers:	
Dornies (GPP)	MUDAS-432
SEL (GPP)	DEC LSI 11
IITB (Werum)	Siemens 310
Uni Karlsruhe (GPP)	Zilog Z80

3. Characteristics and Standardization of PEARL

Basic Concept of PEARL

- Algorithmic language concept mainly based on ALGOL 68 and also influenced by PASCAL
- Syntax adapted to PL/1 where possible
- I/O structure allows to deal with a large variety of virtual devices (called DATIONs) and communication methods
- Abstract data types (like TIME, DURATION, and BIT-string) needed for realtime application

Principle Layout of a PEARL program

MODULE (module name);

SYSTEM;

- Description of hardware configuration (connection)
- Introduction of freely chosen names for I/O terminals and signals

PROBLEM;

Description of actions to be executed, practically independend of environment, e.g.

- Declaration/Specification of data, tasks, procedures
- Start conditions (schedules) of tasks

MODEND;

Characteristics of PEARL

Algorithmic Features

- Block structured language with scoping rules for objects known from the ALGOL family
- Comprehensive choice of data types and possible attributes
- Compound type STRUCTURE (similar to the PASCAL record)
- Communication between processes via GLOBAL objects with optional assignation protection
- PROCEDURE declarations with parameter transfer call-by-value or call-by-reference
- Definition of abstract data types by using TYPE and OPERATOR declaration

Realtime Features

- o Parallel processing by using TASK blocks and PRIORITY
- o Declaration of events (INTERRUPT and SIGNAL)
- o Synchronization of parallel activities by SEMAs and BOLTs

Input and Output Features

- o Data-stations (DATIONs), generalizing real or virtual peripherals or I/O channels
- o Interfaces, mapping data-stations with different properties onto each other to offer the possibility to define formatting routines (objects of type CONTROL)
- o An object of type DATION represents in general a set of one to four channels:
  - Data channel (transfers values of PEARL objects)
  - Control channel (transfers values of type CONTROL)
  - Interrupt channel (signals events of type INTERRUPT)
  - Signal channel (signals event of type SIGNAL)

Standarization of PEARL

- o DIN (German Institute for Norm Control, Berlin) is given the task to care for a PEARL language description
- o Means to describe semantics of realtime features of programming languages are based on Petri Nets
- o DIN Draft Standard for Basic PEARL written and published in June 1978
- o DIN Draft Standard for Full PEARL is scheduled for late 1979

Basic PEARL is the common and minimal subset of Full PEARL which each implementations must contain. Some of the (Full) PEARL features Basic PEARL does not have are:

- Objects of type Reference (pointers) and BOLT
- Extensibility (new data types and operators)
- LABEL variables
- Multi-dimensional STRUCTUREs
- Nested tasks (subtasks)
- Dynamic priority change
- Interfaces
- Graphic I/O

Basic PEARL has been submitted by DIN to the international working group TC97/SC5/WG1 "Programming languages for the Control of Industrial Processes (PLIP)" of ISO (International Organization for Standardization) as candidate for ISO standard. Full PEARL is to follow in due time.

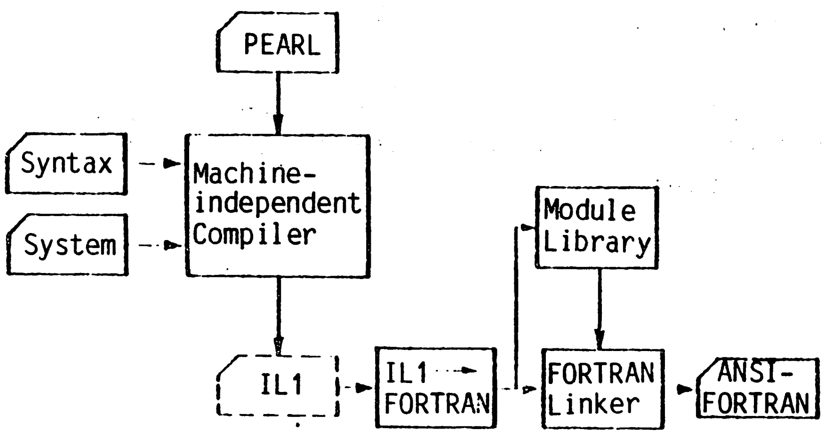
4. Experience with PEARL

- At present about 130 PEARL systems either in operation (the majority) or being installed (approx. 100 systems are truly industrial)
- Quantitative results published (up to 80 % savings)

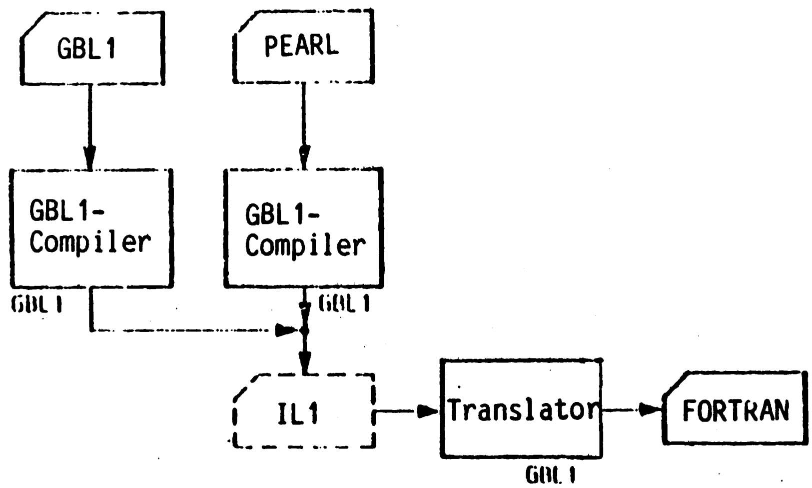
Realtime application area	Number of systems
Metallurgical plants, rolling mills	37
Power distribution	31
Power generation	4
Prime materials, chemical	16
Water supply services	10
Other public services (television, traffic, Spacelab etc.)	7
Other industrial	9
Warehouse, storage	7
PEARL R+D and education	15

5. PEARL on HP 3000

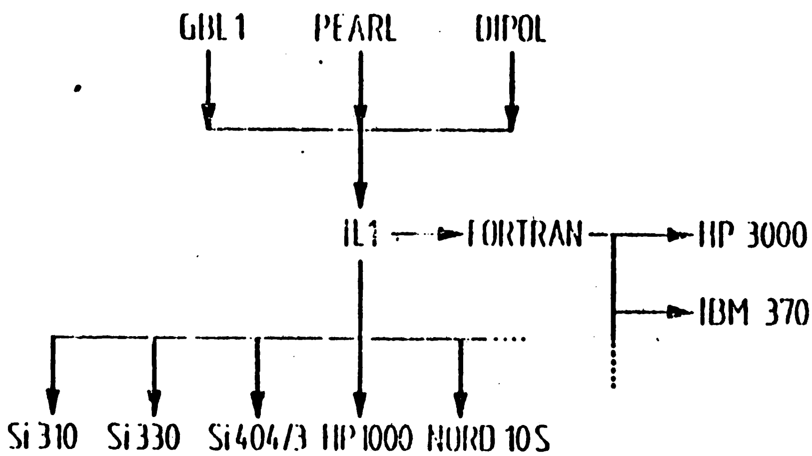
- o PEARL subset on HP 3000 is Basic PEARL
  + multi-dimensional STRUCTUREs
  + objects of type Reference and BOLT
  + declaration of data types and operators
- o Machine-independend compiler
- o Control of syntax analysis by a list to minimize expenses by changes of the language syntax
- o Control of SYSTEM division interpretation by a list written in PEARL
- o Runtime routines using the HP 3000 FORTRAN-library
- o Universal PEARL operating system written in GBL1 (PL/1-subset)
- o PEARL operating system implemented on MPE/II and MPE/III without modifications of the MPE operating system
- o Three HP 3000 processes running for PEARL
  - one process containing all PEARL tasks
  - one process realizing I/O operations
  - one process for timing control
- o five software interfaces betweeen MPE and the PEARL operating system
- o Code generator to produce HP 3000 RBMs instead of FORTRAN announced



Principle of the PEARL compiler system on HP 3000

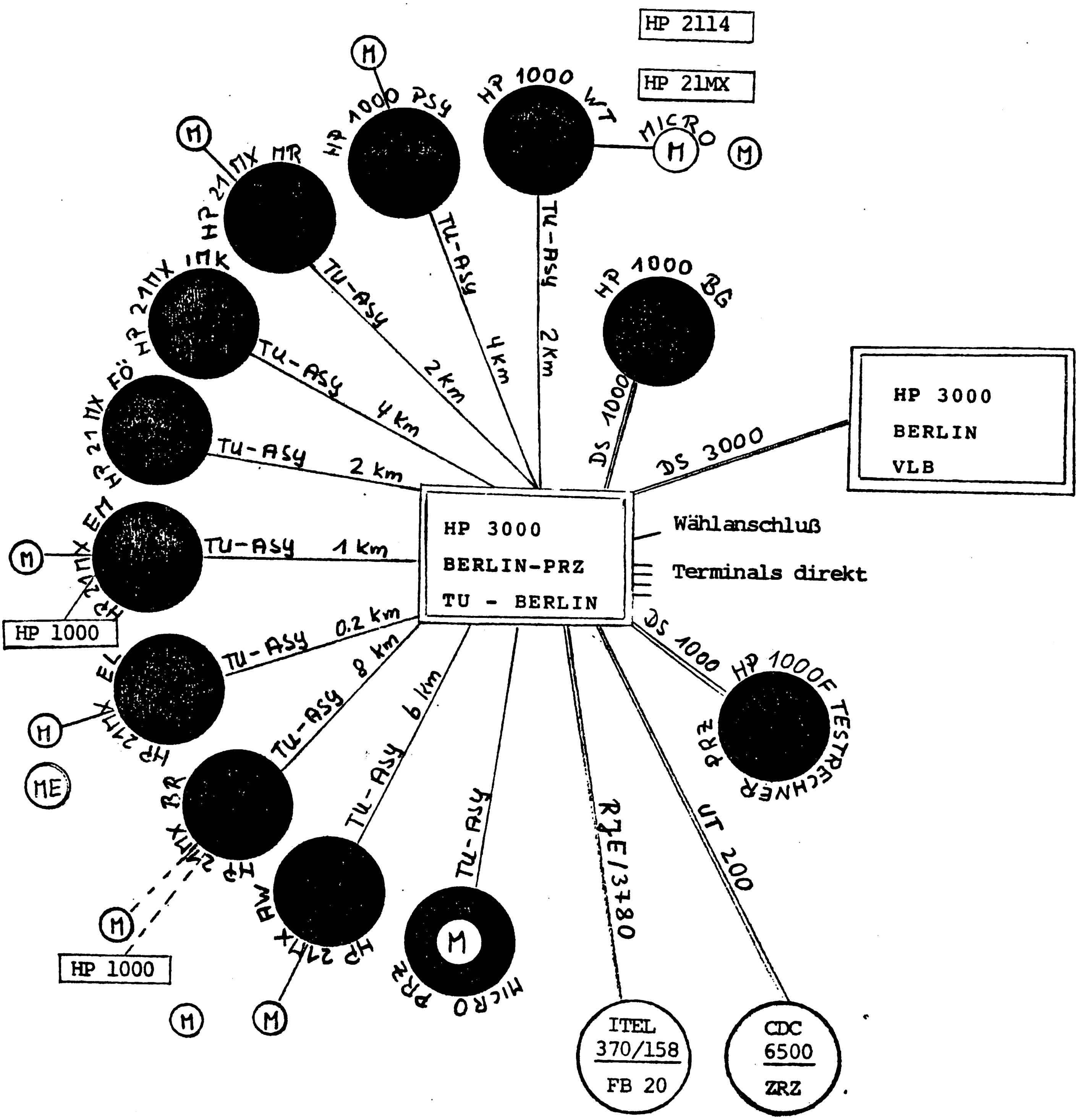


Method of implementing the PEARL compiler system on HP 3000



Advantage of using machine independent compilers by generating IL1 (Intermediate language) as computer output

6. PEARL on a Computer Network HP 3000 - HP 1000



The Realtime-Computer-Network of the Technical University Berlin consists of 11 realtime-systems HP 21MX and HP 1000 in star-configuration with a central computer HP 3000. The satellites are used in various fields of process control like psychology, ergonomics or brewery and can have micro-computers as subcomputers. The central HP 3000 is mainly used for program development and remote file access. In order to increase the availability of the system a project has been started in 1978 to introduce PEARL into the process computer network.

The main characteristics of this PEARL implementation are:

- Introducing PEARL into the process-computer-network is done in three phases:
  - Implementing the excisting PEARL system on the central HP 3000
  - Implementing a cross-compiler system for HP 1000 on HP 3000
  - Implementing a stand-alone compiler system on HP 1000
- For all three phases the same compiler main part is used; a code generator is written to produce HP 21MX relocatable code
- Sophisticated DS-features will be included to make PEARL applicable for realtime-computer-networks
- As a prospect it is planed to implement PEARL cross software on HP 3000 for microprocessors to be configured a subsystem under each satellite computer

References:

T. Martin: Realtime Programming Language PEARL - Concepts and Characteristics
Proceedings of COMPSAC 78 page 301

Full PEARL Language Description,
PDV-Report KfK-PDV 130,
Kernforschungszentrum Karlsruhe GmbH, 1977

DIN 66253 Basic PEARL Draft Standard
Beuth Verlag GmbH, Berlin, Cologne 1978

K.Rebensburg: Real-time-computing with the Computer network of the technical university Berlin

Example of a simple PEARL application

A pipeline shall be pressure controlled and is properly instrumented for this purpose (Fig. 1).

Fig. 2 shows the function/data block structure of the planned computer control system. The next step would be to choose a hardware configuration and connect the sensors, lamp, and control devices to certain clamps of the computer I/O devices (Fig. 3).
There are two tasks: Task 1 controls the pressure by means of a simple on/off algorithm. The valve is opened and closed by outputting bit pattern "10" or "01", respectively. Task 2, in addition, takes measures to alarm and quickly release the pressure, if a certain high limit is reached. Next the PEARL program can be implemented. (In the following text capital letters are used for designating PEARL language elements.) Before this, the general program structure of PEARL must be explained. PEARL programs are composed of independently compilable units, the so called MODULEs. Connections between MODULEs are established through GLOBAL objects. In general, a MODULE consists of a part describing the hardware configuration (especially the process I/O connections), called SYSTEM division, and another one containing the formulation of the problem, called PROBLEM division:

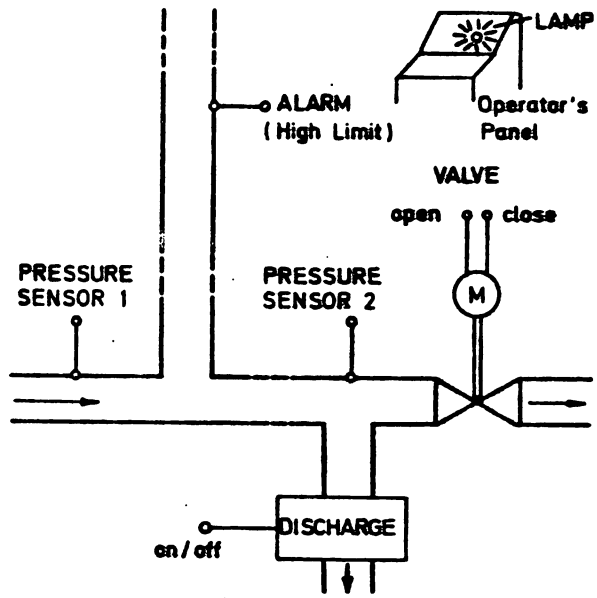


Figure 1 Pressure control scheme for pipeline

```
MODULE (module name);
    SYSTEM;
        - Description of hardware
          configuration (connections)
        - Introduction of freely chosen
          names for I/O terminals (DATIONs),
          interrupts, and signals
    PROBLEM;
        Description of actions to be executed,
        practically independent of environ-
        ment, e.g.
        - Declaration/Specification of
          data, tasks, procedures
        - Start conditions (schedules)
          of tasks
MODEND;
```

The purpose of this separation is the following: The PROBLEM division which contains the algorithmic and organizing part of the program becomes invariant against hardware changes. Thus, portability of the control program is achieved!
The PROBLEM division is structured by means of TASKs and PROCEDUREs. TASKs are independently and simultaneously runnable program parts. PROCEDUREs (not used in the example) are, as usual, dependent on the CALLing program part.

With this information the reader should be almost able to understand the PEARL program, given in Fig. 4, utilizing the self-documentation feature of PEARL. In the SYSTEM division the transfer direction is defined by arrow symbols. Note how the user names of the devices (i.e. DATIONs) are introduced like labels. The other hardware names are system names known by the compiler.

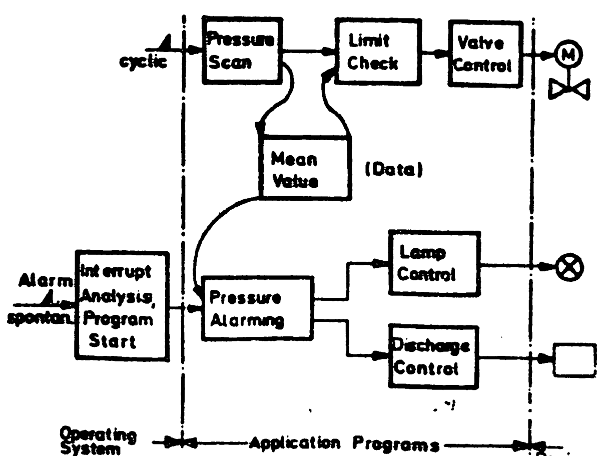


Figure 2 Function/Data block structure of computer control system

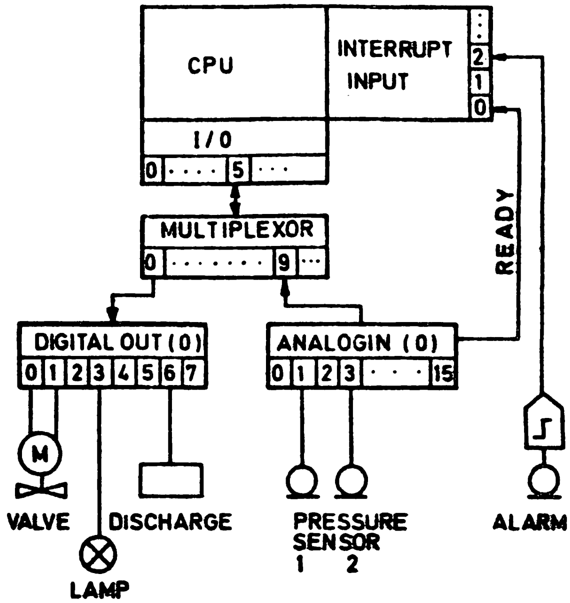


Figure 3 Computer hardware configuration with process connections

```
MODULE; /*EXAMPLE PRESSURE CONTROL*/
SYSTEM;

MULTIPLEXOR    <—>    CPU*5;
DIGITALOUT(0)  <—     MULTIPLEXOR*0;
ANALOGIN(0)     —>    MULTIPLEXOR*9;

VALVE      :      <—    DIGITALOUT(0)*0.2;
LAMP       :      <—    DIGITALOUT(0)*3.1;
DISCHARGE  :      <—    DIGITALOUT(0)*6.1;

PRESSURESENSOR1:   —>    ANALOGIN(0)*1;
PRESSURESENSOR2:   —>    ANALOGIN(0)*3;

READY:   —>   INTERRUPTINPUT*0;
ALARM:   —>   INTERRUPTINPUT*2;

PROBLEM;
SPECIFY  VALVE DATION OUT BASIC DIM() TFU,
         (LAMP, DISCHARGE) DATION OUT BASIC,
         (PRESSURESENSOR1, PRESSURESENSOR2) DATION IN BASIC DIM() TFU,
         ALARM INTERRUPT;

DECLARE (PRESS1, PRESS2, MEANPRESS) FLOAT,
        OPENVALVE BIT(2)  INITIAL('10'B),
        CLOSEVALVE BIT(2) INITIAL('01'B),
        (TURNON, TURNOFF) BIT INITIAL('1'B,'0'B);

START:           TASK GLOBAL; /*ACTIVATED FROM OUTSIDE*/
                   :
                 EVERY 1 SEC ACTIVATE PRESSURECONTROL;
                 WHEN ALARM ACTIVATE ALARMING;
                 END; /*OF TASK START*/

PRESSURECONTROL: TASK PRIORITY 5;
                 TAKE PRESS1 FROM PRESSURESENSOR1;
                 TAKE PRESS2 FROM PRESSURESENSOR2;
                 MEANPRESS = (PRESS1 + PRESS2)/2;
                 IF MEANPRESS >= 30
                   THEN SEND OPENVALVE TO VALVE;
                   ELSE IF MEANPRESS <20 THEN SEND CLOSEVALVE TO VALVE;
                        FIN;
                 FIN;
                 END; /*OF TASK PRESSURECONTROL*/

ALARMING:        TASK PRIORITY 3;
                 SEND TURNON TO LAMP;
                 SEND TURNON TO DISCHARGE;
                 WHILE MEANPRESS >= 20
                   REPEAT;
                   AFTER 2 MIN RESUME;
                 END; /*OF LOOP*/
                 SEND TURNOFF TO LAMP;
                 SEND TURNOFF TO DISCHARGE;
                 END; /*OF TASK ALARMING*/
MODEND;
```

Figure 4 PEARL source program

Collection of PEARL Features and Examples

A1. PEARL OBJECTS (DATA TYPES)

(1) PROBLEM DATA

Integer	FIXED (Precision)
Floating Point	FLOAT (Precision)
Bit String	BIT (Length)
Character String	CHAR (Length)
Time Interval	DURATION
Point in Time	CLOCK

(2) DATA FOR PROCESS CONTROL

Task	TASK
Interrupt } Events	INTERRUPT
Signal } Events	SIGNAL
Semaphore	SEMA
Bolt	BOLT

(3) DATA FOR I/O CONTROL

File, Device } Data Station	DATION
Standard Formats	F, E, A, B, T, D, LIST, X, SKIP, PAGE, etc.
C-Channel Value	CONTROL
Remote Format	FORMAT
User-Defined Format	INTFAC

(4) DATA REFERENCE

Pointer	REF

(5) COMPOUND OBJECTS

Array	Ident. (bound-pair-list)
Structure (Data Hierarchy)	STRUCTURE

(6) USER-DEFINED DATA TYPES TYPE

A2. EXAMPLES OF PEARL OBJECTS

```
DECLARE I FIXED,
        (F,G) FLOAT,
        B BIT (16) INIT ('OOFF'B4),
        C CHAR (4) INIT ('ABCD'),
        A (1:10,-5:5) FLOAT,
        RF REF FLOAT,
        TAG LABEL RANGE(M1, M2),
        PIECE STRUCT
            [NR FIXED (10),
             NAME CHAR (8),
             DATE STRUC
                [(DAY, MONTH, YEAR) FIXED (2)
                ]
            ];
```

```
        ┌─NR
        ├─NAME
PIECE ──┤
        │                  ┌─ DAY
        └─DATE ────────────┼─ MONTH
                           └─ YEAR
```

Component access: PIECE.NAME
PIECE.DATE.YEAR
(String Selection similiar)

A3. TRANSFER-OF-CONTROL STATEMENTS

Block	BEGIN; Local Declarations Statements END;
Conditional-Statement (Alternative)	IF Bit-one-Expression THEN Statements 1 ELSE Statements 2 FIN;
Case-Statement (Selection)	CASE Expression ALT Statements 1 . . . ALT Statements N OUT Statements (Error Exit) FIN;
Repeat-Statement (Iteration)	FOR Identifier (Loop Index) FROM Expression (Initial Value) BY Expression (Step Width) TO Expression (End Value) WHILE Bit-one-Expression REPEAT Local Declarations Statements END;

A4. TASKING AND SCHEDULING

TASK DECLARATION

```
T:  TASK PRIO I;
    Local Declarations
    Statements
    END;
```

TASK EXECUTION

at once	ACTIVATE T; ACTIVATE T PRIO (3*I);
at point in time	AT 8:30:0 ACTIVATE T; DCL TIME CLOCK, PERIOD DUR; . . . TIME = 9:0:0; PERIOD = 2 HRS; . . .
periodically	AT TIME ALL PERIOD DURING 12 HRS ACTIVATE T;
at interrupt	DCL FULL INTERRUPT; . . . WHEN FULL ACTIVATE T;
after 10 sec	WHEN FULL AFTER 10 SEC ACTIVATE T;
periodically for 5 min	WHEN FULL ALL 2 SEC DURING 5 MIN ACTIVATE T;
Delay for Time Span	AFTER 5 SEC RESUME;
Delay till Interrupt	WHEN MESSAGE RESUME;
Temporary Stop	SUSPEND; or SUSPEND T;
Continue	CONTINUE T PRIO 3;
Kill Activity	TERMINATE; or TERMINATE T;
Remove all Schedules Pending	PREVENT T;

Note: Appendix A and B are taken form "Realtime Programming Language PEARL-Concepts and Characteristics" by T. Martin (see references).