

# **WP9 - Process control application requirements and platform analysis**

**Deliverable D9pc.1**

**WP9 - Process control application requirements and platform analysis: Deliverable  
D9pc.1**

by Ales Hajny and Stanislav Benes

Copyright © 2003 by Ocera

# Table of Contents

<b>1. Introduction .....</b>	<b>1</b>
1.1. Document Structure.....	1
<b>2. The process control application .....</b>	<b>2</b>
2.1. The real application .....	2
2.2. The modified application - platform analysis .....	2
2.3. The control algorithm for turbocompressor .....	3
2.4. The control algorithm for turbine .....	4
<b>3. Development plan .....</b>	<b>6</b>
3.1. 1. Configuration of system for verification.....	6
3.2. 2. Block diagram of process node .....	6
3.3. 3. Developed components.....	7
<b>4. Verification plan.....</b>	<b>9</b>
4.1. Developed components verification .....	9
4.2. OCERA components validation.....	10

# Chapter 1. Introduction

## 1.1. Document Structure

This document consists of three parts:

- **Process control application**, describing control algorithms which will be implemented in the control station.
- **Development plan**, which is requirement specification referring to the control station and to the simulator.
- **Verification plan**, which states methodology of evaluation of real time behaviour of the system.

# Chapter 2. The process control application

## 2.1. The real application

The high-pressure gas turbocompressor plant in Kralice (the Czech Republic) is a real process control application running under RTOS OS9. The plant serves for high pressure gas transport in the main gas pipeline from Russia to Germany.

Machines for gas transport consist of a gas turbocompressor and a gas turbine as a drive. The gas turbocompressor is a centrifugal machine on a common shaft with low-pressure part of the gas turbine. It has an independent oil system for shaft stuffing with two pumps.

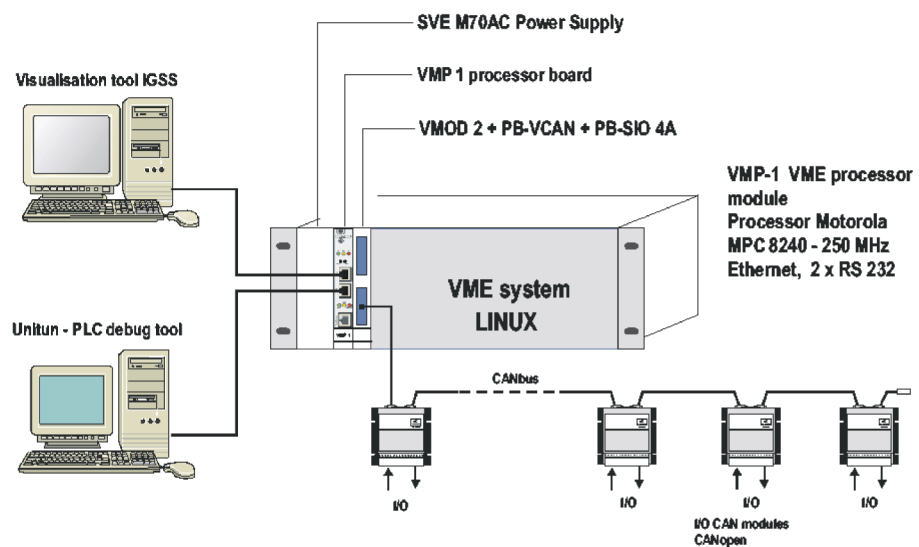
## 2.2. The modified application - platform analysis

A subset of real control algorithms was chosen for OCERA components validation. These algorithms will run on a test control station equipped with PowerPC processor MPC 8240 under OCERA RTLinux (see Figure 2.1).

The gas turbocompressor technology will be simulated on a technology simulator equipped with MSM586SEV processor (PC104 family) under OCERA RTLinux too (see Figure 2.2).

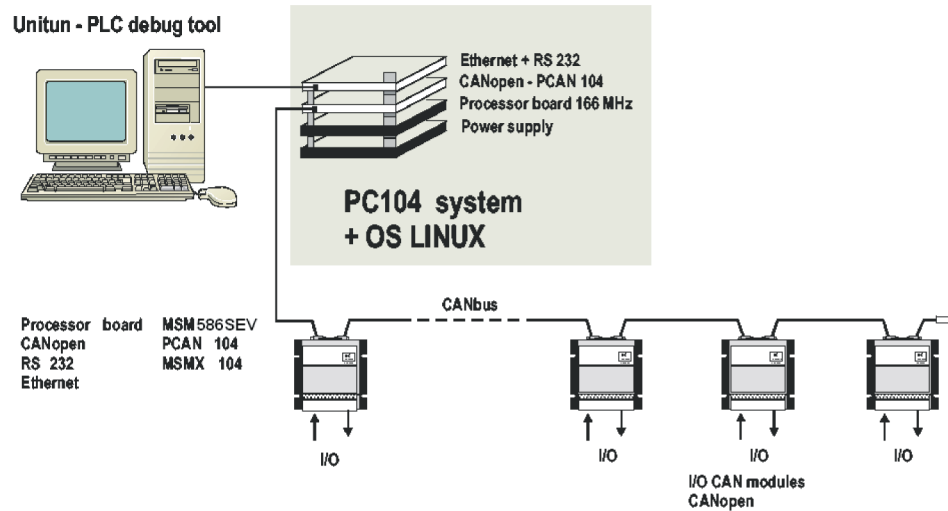
**Figure 2.1**

### System Configuration – Control System



**Figure 2.2**

## System Configuration Technology UniTun Simulator



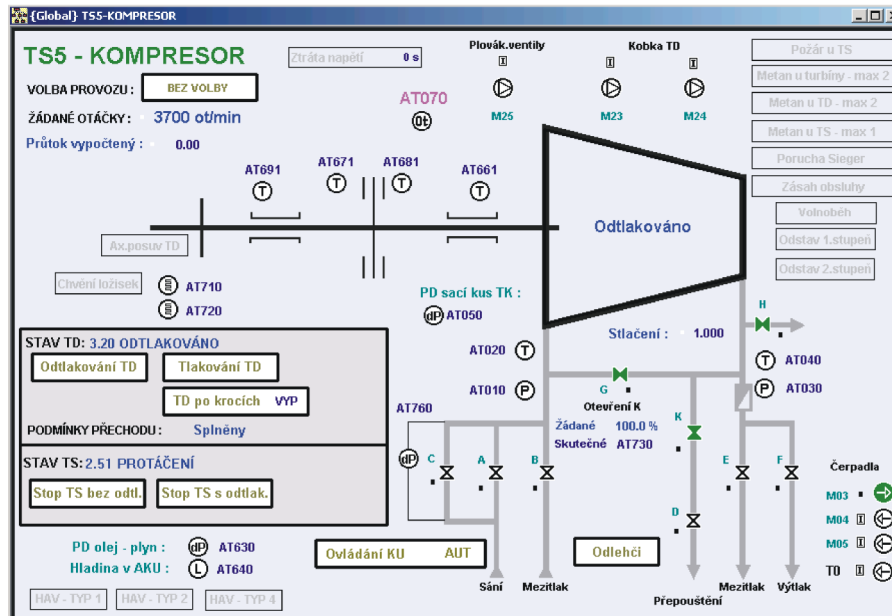
### 2.3. The control algorithm for turbocompressor

There is the sequence of the technology states of the turbocompressor needed for its pressuring (see Figure 2.3).

1. **Lubricant initiation**, the oil pump M03 is switched on to initiate lubrication.
2. **Oil stuffing initiation**, the oil pump T0 is switched on to initiate the stuffing of the shaft.
3. **Open bypass valve C**, the valve C is opened for gas blowing-through. It is necessary to remove the explosive mixture of gas and air.
4. **Blowing-through takes 15s**, the blowing-through must take 15 s at least.
5. **Close valve H**, the blowing-through is finished.
6. **Connect the turbocompressor to the pipe**, the valves A, D, K are opened and the valve C is closed to connect turbocompressor to the pipe.
7. **Pressured**, the turbocompressor is pressured with gas, machines are ready for start of the turbine.

An automatic execution of the described sequence is initiated by pushing the button "Tlakovani TD" in Figure 2.3.

**Figure 2.3**



## 2.4. The control algorithm for turbine

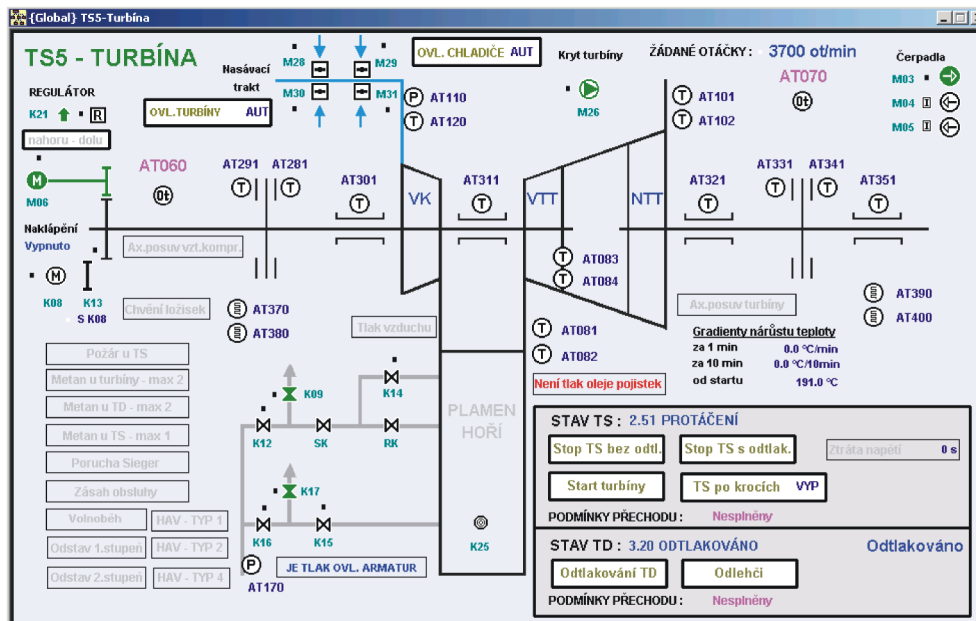
There is the sequence of the technology states of the turbine necessary to prepare it to operating state (see Figure 2.4).

1. **Test of start conditions**, the starter M06 and the ignition K25 are switched off, the telescope device K13 is put out, the hydromotor K08 and the oil pump M03 are OK, speed regulator K21 is UP, the burners K14, K15 and the valves K12, K16 are closed and the valves K09, K17 are opened.
2. **Turbine ventilation**, switch on the ventilator M26 and open the shutters M28, M29, M30, M31.
3. **Start oil pump**, switch on the oil pump M03 and if the turbocompressor isn't ready, start its algorithm.
4. **Speed regulator K21 DOWN**, close a fuel inlet to minimum with the speed regulator K21 .
5. **Switch on the starter M06**, the starter gives the turbine an initiate speed.
6. **Push in the telescope device K13**, if the turbine speed is at least 12 rpm the telescope device moves the hydromotor K08 into start position.
7. **Start the hydromotor K08**, to give the turbine higher speed.
8. **Open the valve K16**, if the turbine speed is higher then 100 rpm open K16 as a fuel inlet.
9. **Close valve K17**, if the turbine speed is higher then 300 rpm close K17 as a ventilation fuel outlet.
10. **Switch on the ignition K25**.
11. **Open the burner K15**, to fire a flame in combustor of the turbine and wait 10 seconds.
12. **Open the valve K12**, as another fuel inlet.
13. **Close valve K09**, as a ventilation fuel outlet.
14. **Open the burner K14**, to give the turbine higher speed.

15. **Stop ignition**, switch off the ignition K25, close the valve K16 and the burner K15 and open fuel ventilation K17 and wait 10 minutes to get the turbine hot.
16. **Speed regulator K21 UP**, to open the valve RK for fuel inlet regulation.
17. **Speed regulator K21 STOP**, to stop speed escalation and wait 10 minutes.
18. **Speed regulator UP**, to reach at least speed 3700 rpm.
19. **Close the valve G**, as a turbocompressor bypass for gas.
20. **Speed regulation**, put requested speed of the turbine into the variable "ZADANE OTACKY".

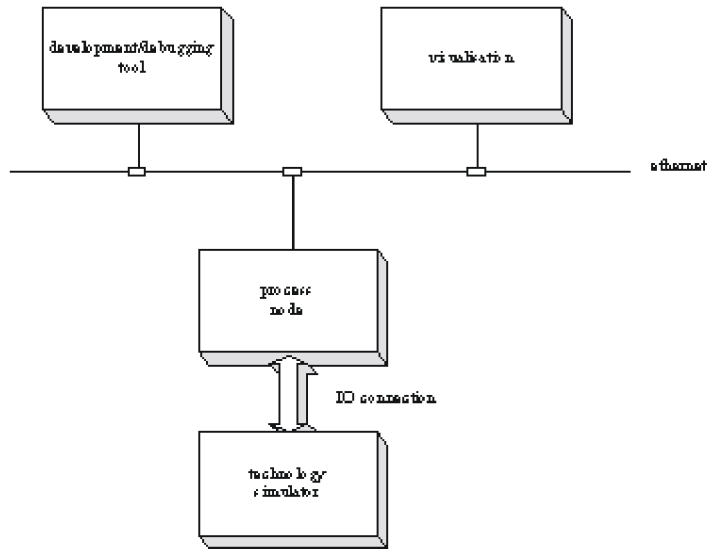
Now the machines are ready to transport gas to the pipe called "Vytlak" see the Figure 2.3.

Figure 2.4

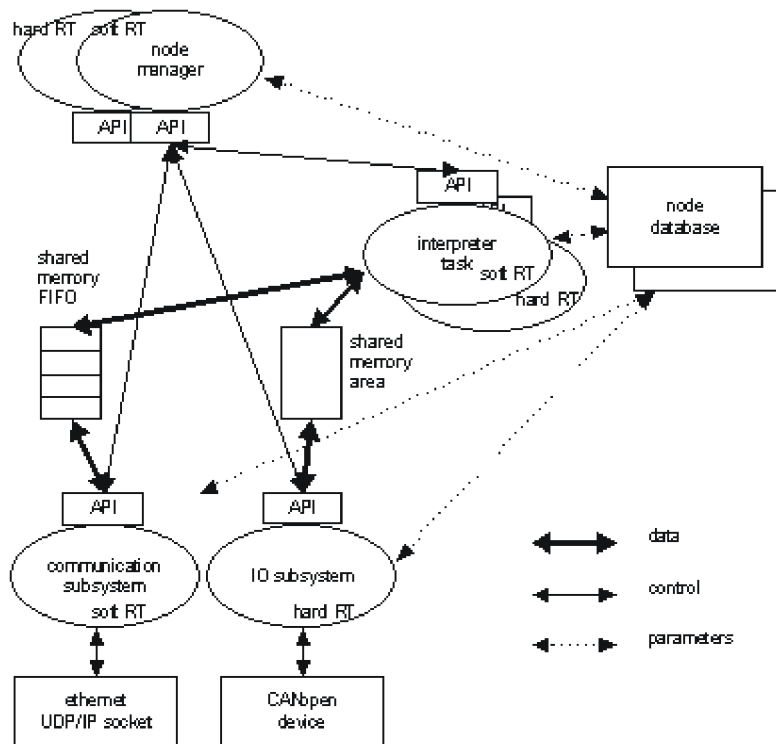


# Chapter 3. Development plan

## 3.1. 1. Configuration of system for verification



## 3.2. 2. Block diagram of process node



## 3.3. 3. Developed components

### □ Node manager

- **Description:** Node manager maintains all information regarding node as control system subject. Node database with interpreted code is maintained here and algorithm interpreter is controlled in terms start, stop interpretation, attach/release IO subsystem and communication subsystem. Configuration/reconfiguration of IO subsystem and ethernet communication subsystem is controlled from here. Node manager will contain hard RT task responsible for accurate node timing and soft RT task responsible for node management.

- RT space placement: both hard RT and soft RT
- SW interface: Node manager API

#### □ **Control algorithm interpreter**

- Description: Control algorithm interpreter uses code, prepared in node database, and performs actions as specified by this interpreted code. Interpreted code is prepared offline in development and debugging tool running on host PC and downloaded into target system. Application control is described in languages according to IEC 61131 and then compiled. Download of interpreted code is controlled by node manager. Control algorithm interpreter contains IO subsystem interface and communication subsystem interface. Interpreted code is divided into interpreted "tasks", which are represented with Linux tasks. For critical control algorithms are these tasks placed in hard RT, the rest is placed in soft RT.
- RT space placement: both hard RT and soft RT
- SW interface: API for interpretation control, Node databases with interpreted code

#### □ **Ethernet communication subsystem**

- Description: Ethernet communication subsystem transports messages via ethernet network using UDP/IP sockets. Subsystem take care for message confirmation, timeout evaluation and communication retries. Subsystem keeps and maintains net topology information, which is obtained with broadcast messages among nodes in topology.
- RT space placement: soft RT
- SW interface: Communication subsystem API library
- HW interface: onboard Ethernet controller on VMP1 board

#### □ **IO subsystem with CANopen communication**

- Description: IO subsystem connects algorithm interpreter with remote IO modules connected to node with CAN bus and protocol CANopen. IO modules provides direct technology control. CANopen protocol is realized by CANopen device and connected to the subsystem with its API library. CANopen subsystem is configured according to node database and can act as CANopen master or slave, as a master can manage several CANopen remote IO nets with attached modules. Used nets and modules with appropriate parameters are specified in database.
- RT space placement: hard RT
- SW interface: IO subsystem API library
- HW interface: PB-VCAN piggyback carried on universal VME IO board VMOD-2D

# Chapter 4. Verification plan

## 4.1. Developed components verification

- **1. Communication subsystem**

- **1.1. Test of communication latency**

- Description: Test application will send message through communication system to other node and will receive confirmation message, time delay will be measured for given period and communication latency statistics will be evaluated. Message length is chosen random from given interval.
- Test environment: Two nodes, PC node and embedded system node with VMP1, both with RTLinux, connected via ethernet

- **1.2. Test of communication throughput**

- Description: Test application will send message through communication system to other node and will receive confirmation message, amount of sent data will be counted for given period and data throughput will be evaluated. Message length is chosen random from given interval
- Test environment: Two nodes, PC node and embedded system node with VMP1, both with RTLinux, connected via ethernet.

- **1.3. Test of communication stability**

- Description: Test application in two nodes will be sending message through communication system to each other will receive confirmation message. Test will be running for reasonably long period (several days) and all problematic behavior will be logged and analyzed.
- Test environment: Two nodes, PC node and embedded system node with VMP1, both with RTLinux, connected via ethernet.

- **2. Interpreter of algorithms**

- **2.1. Test of interpretation capacity**

- Description: In node with control algorithm interpreter will be downloaded test applications containing selected sets of interpreted elements. For each application time consumption will be measured and time consumption per one element will be evaluated.
- Test environment: One embedded system node with VMP1 running RTLinux, PC with diagnostics/debugging tool.

- **2.2. Test of interpretation accuracy**

- Description: In node with control algorithm interpreter will be downloaded test application containing all interpreted elements. This test application will contain self checking coconnections with results available in development/debugging station.

- Test environment: One embedded system node with VMP1 running RTLinux, PC with diagnostics/debugging tool.
- **2.3. Test of IO connections**
  - Description: In node with control algorithm interpreter will be downloaded test application containing all available IO connected to application. IO connections will be connected in loopbacks from outputs to inputs and I and O values will be compared by test application. Results will be available in development/debugging tool.
  - Test environment: Testing rack with embedded system node with VMP1 running RTLinux, remote IO modules, PC with diagnostics/debugging tool.
- **2.4. Test of IO connections latency**
  - Description: In node with control algorithm interpreter will be downloaded test application containing all available IO connected to application with application loopback from inputs to outputs. Inputs will be set by external device and time delay to corresponding output activation will be measured by oscilloscope.
  - Test environment: Testing rack with embedded system node with VMP1 running RTLinux, remote IO modules, oscilloscope, PC with diagnostics/debugging tool.
- **2.5. Test of inter-node communication**
  - Description: Two nodes with control algorithm interpreter will be connected via ethernet and both will contain test application with external variables assigned to each other. Exported variables will be compared by test applications and results will be available in development/debugging tool.
  - Test environment: Two nodes, PC node and embedded system node with VMP1, both with RTLinux, connected via ethernet

## 4.2. OCERA components validation

- **1. Validation of CANopen device**
  - **1.1. Test of communication latency**
    - Description: Test process using CANopen communication will send object to the remote O module and sets direct output pin on VDOUT module. Time delay to the remote O reaction will be measured by oscilloscope. Analogically, time delay for I module object transport to the test process will be measured.
    - Test environment: Testing rack with embedded systems with VMP1 and PC104, both running RTLinux, remote IO modules, oscilloscope.
  - **1.2. Test of communication capacity**
    - Description: Test process using CANopen communication will send as many objects to the remote IO modules and will measure amount of data send per given time interval. Transferred object rate will be evaluated.

- Test environment: Testing rack with embedded systems with VMP1 and PC104, both running RTLinux, remote IO modules.
  
- **2.Validation of EDS parser and CAN/CANopen analyzer**
  - **2.1. Test of analyzer accuracy**
    - Description: Test process using CANopen communication will send given sentence of objects to the remote IO modules and communication on CAN bus will be captured by CANopen analyzer. Captured sentence will be compared to the template.
    - Test environment: Testing rack with embedded systems with VMP1 and PC104, both running RTLinux, remote IO modules, PC with CANopen analyzer.
  
  - **2.2. Test of EDS parser**
    - Description: Template EDS containing all available IO modules will be prepared and analyzer output will be captured and compared to the template.
    - Test environment: Testing rack with embedded systems with VMP1 and PC104, both running RTLinux, remote IO modules, PC with CANopen analyzer.
  
- **3.Validation of interrupt and scheduling latency**
  - **3.1. Interrupt latency measurement**
    - Description: Test interrupt service routine will be installed on internal timer vector. Timer will be set and started by test process, which will evaluate latency statistics. Second independent timer will be used for time measurement.
    - Test environment: Embedded systems with VMP1 and PC104, both running RTLinux.
  
  - **3.2. Scheduling latency measurement**
    - Description: Test process on given priority will be running together with test process, which will hand over activity to the first one. Internal timer will be used for latency statistics evaluation.
    - Test environment: Embedded systems with VMP1 and PC104, both running RTLinux.