# Experimenting a Real-Time Industrial Messaging Service for a Power Utility: TASE.2 over the real-time micro-kernel pSOS+

E. Becquet, becquet@cnam.fr

E. Gressier-Soudan, (author for correspondence), gressier@cnam.fr

Cnam, Laboratoire Cedric, 292 rue St Martin, 75141 Paris Cedex 03, France

Tel : 33 1 40 27 22 96, Fax : 33 1 40 27 22 96

E. Le Grand, erwan.legrand@edf.fr,

EDF R&D, EP/CCC/GA, 6 Quai Watier, 78401 Chatou Cedex, France

and

EDF Centrale de Nogent-sur-Seine, 10 Nogent-sur-Seine, France

G. Hellack, hellack@ghf-automation.de,

*GHF automation GmbH, Tech.-Park Herzogenrath Kerserstrasse 100, D-52134 Herzogenrath, Germany

Preferred Topics : Factory Communications,

more precisely, the paper deals with a protocol for Utility data exchanges,

# Experimenting a Real-Time Industrial Messaging Service for Power Utility: TASE.2 over the real-time micro-kernel pSOS+

E. Becquet[+], E. Gressier-Soudan[+], E. Le Grand[$!], G. Hellack[*]

becquet@cnam.fr, gressier@cnam.fr, erwan.legrand@edf.fr, hellack@ghf-automation.de

[+]Cnam, Laboratoire Cedric, 292 rue St Martin 75141 Paris Cedex 03, France

[$]EDF R&D, EP/CCC/GA, 6 Quai Watier, 78401 Chatou Cedex, France

[!]EDF Centrale de Nogent-sur-Seine, 10 Nogent-sur-Seine, France

[*]GHF automation GmbH, Tech.-Park Herzogenrath Kerserstrasse 100, D-52134 Herzogenrath, Germany

## Abstract

Our project aimed to evaluate the ability of the TASE.2 (Telecontrol Application Service Element version 2) standard to support EDF's communication requirements for small and medium production units. EDF is the French power supplier. Our work focused on TASE.2 conformance blocks 1, 2 and 5. TASE.2 is a companion standard of ISO MMS. We studied TASE.2 services through the port of an off-the-shelf product over pSOS+ a real-time kernel. This paper describes our tests and our results. The port has been successful and our implementation behaves like the reference implementation over Windows NT. Finally, this work allowed us to investigate the TASE.2 standard, and contributes to define functional tests.

## Key words :

TASE.2, ICCP, pSOS+, MMS, real-time, tests, conformance.

## Note to the reviewers :

This work has been granted by EDF R&D. It has been done under certain constraints. We can't give the name of the product that we tested. We can't give implementation details nor performance results. Such information is covered by a restricted access agreement.

These requirements are a drawback to write a paper, but we believe that our study is valuable. Despite an important work related to code development, strictly speaking, our contribution is not in the area of implementation. Our results focus essentially on a practical experiment with TASE.2, a deep investigation inside the standard and in defining functional testing. Without any standard tools, our tests offer a conformance evaluation of a product under a constraint of interoperability. These results were not the original goals of the study, we try to show this aspect in the beginning of the paper.

# 1. Introduction

Our project aimed to evaluate the ability of the TASE.2 (Telecontrol Application Service Element version 2) standard [IEC96a][IEC96b][IEC97][KEM96] to support EDF's requirements [LEG00]. EDF is the main french power utility. TASE.2 is also known as ICCP (Inter-Control Center Protocol). TASE.2 is a candidate technology to support the exchange of power production data between dispatching centres and production units.

TASE.2 is a companion standard of the popular ISO MMS (Manufacturing Message Specification) [VAL92]. TASE.2 brings ISO MMS back in the front of the scene in the context of Utilities (power, gas, water…). Therefore, the old MMS technology empowered with TASE.2 features is becoming a brand new solution, and can be the basis of a wide information system able to support distributed control applications for utilities !

Our work focuses on TASE.2 conformance blocks 1, 2 and 5. Block 1 deals with periodic data exchanges. Block 2 deals with event-based data exchanges. Block 5 deals with remote control of devices. TASE.2 adds real-time management features to variable exchanges. These features meet power production management real-time constraints. All these needs related to real-time requirements lead EDF to select a real-time kernel as a basis for power data management applications. The pSOS+ real-time kernel, evaluated in a previous study [BEC01a] has been chosen. pSOS+ is supplied by Wind River, (formerly from ISI), it is dedicated to embedded systems.

The overall study hereafter describes our experiment with the port of an off-the-shelf TASE.2 server product over pSOS+, and its evaluation towards the TASE.2 standard. TASE.2 is an emerging technology, and testing new products is an important challenge for the utility industry. The port has been successful and our implementation behaves like the reference implementation over Windows NT. The result of this study is not only a successful port of a product. Indeed, it is a deep investigation inside the TASE.2 standard leading our work close to conformance testing without the help of any conformant product and under a requirement of interoperability because the client was a different product from a different supplier. Finally, this appears to be an un-expected result, and is in fact our most significant outcome. We can assert that the goals of the study evolved smoothly and unconsciously during the work. This had an important impact on our test methodology. We evolved from a black box approach toward an open box approach. The black box approach avoids the knowledge of implementation details. The open box approach allows to change variable values and to observe the behavior of the implementation. Specific tools have been developed to satisfy the required fine grain testing.

The paper is organized as follow. Section 2. presents the context of this work. Section 3. gives an overview of TASE.2, section 4 describes variable objects, and section 5 describes device objects. Section 6 is dedicated to our port of the product on the real-time kernel. Section 7 is dedicated to our test approach and the test platform. Section 8 gives the key point of functional testing. Section 9 concludes our study.

## 2. Context of our TASE.2 evaluation

The TASE.2 technology study is incorporated into a larger project that deals with Java for industrial applications in our research lab. More precisely, the aim of this project is to evaluate the efficiency of RT-Java for process control with light real-time constraints.

We are implementing a Java Based Embedded Remote Monitoring Tool for Small and Medium Power Plant Units [BEC01b][REV01]. The overall architecture of this application is given in Figure 1.

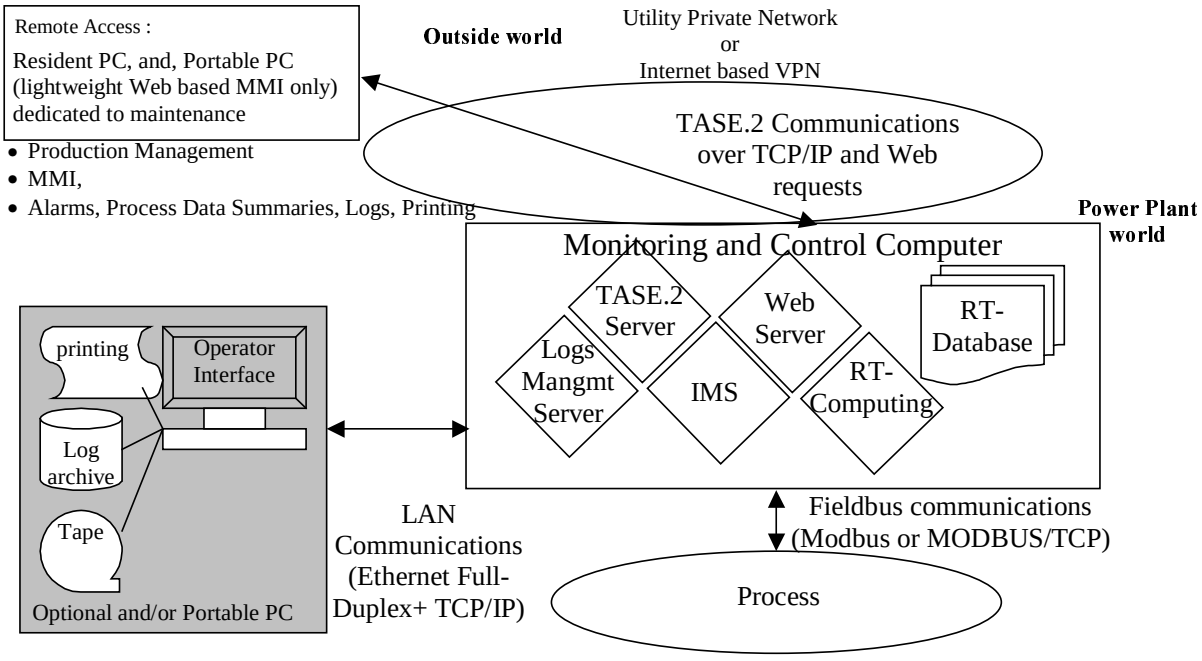


**Figure 1.** Architecture of a Monitoring and Control application for small and medium power control units

A monitoring and control station can be accessed remotely by production management computers, operators or maintenance crew. Remote control computers implements: production management functions, advanced man/machine interface (MMI), alarms, process variables and objects, logs and printing functions. Maintenance crew uses portable PC with lightweight web based monitoring and control operator interface [GUI00]. The monitoring and control station is a real-time computer (in our case, a MBX 821, a Power PC processor) with a real-time kernel, pSOS+. It supports

real-time computing, the instrumentation and measurement system (IMS), a real-time database with the process data and production data for dispatching centers, TASE.2 communications servers and a tiny web server.

This station uses a fieldbus system to communicate with sensors and actuators. The fieldbus system can be classic or Ethernet/TCP/IP based (industrial Ethernet). The Ethernet LAN is full duplex to allow a better determinism in communications. If required, the LAN can be used to interact with a PC that supports input/output devices. A point-to-point link enables access to/from outside world. It can be a TCP/IP based dedicated leased line, or an Internet based virtual private network. It supports TASE.2 communications and web requests.

Currently, only some of the software components are implemented in Java [REV01]. A real-time Java Virtual Machine, called PERC [NIL96] allows a deterministic behavior of Java programs. PERC supports a web server implemented as a Java servlet server. An hundred of handled objects are supposed to be monitored, we expect that the real-time computing unit could be fully realized in Java. A MODBUS/TCP client in Java exists and can be downloaded from Schneider Electric [MOD01], it features the fieldbus system. The log server can be written in Java. Access to native code is allowed using Java Native Interface (PERC Native Interface in our case).

## 3. TASE.2 general overview

The TASE.2 protocol [IEC96a][IEC96b][IEC97][KEM96], also known as Inter-Control Centre Communications Protocol (ICCP), allows data exchange between utility control centres and production units. It may be applied in any domain with same requirements. When the TASE.2 standard specifications are examined, it can be asserted that TASE.2 is a general-purpose real-time application protocol that could be used for factory automation. This is especially true in the current context where industrial Ethernet (Full Duplex Ethernet plus TCP/IP) brings new opportunities for fieldbus systems [papier tendance sur les bus de terrain].

### 3.1. TASE.2 communication stacks

The TASE.2 protocol runs over the MAP protocol stack on top of the popular Manufacturing Message Specification (MMS) ISO standard [VAL92]. To avoid some drawbacks blamed to MMS, like the complexity of its implementation, its costs, MMS can be used over TCP/IP using the RFC 1006 [ROS87] in place of the full OSI stack. In that case, layers 4, 5, 6 and 7 from OSI are involved.

MMS defines an abstract representation of a device (designated by Virtual Manufacturing Device or VMD). VMDs store and handle 12 abstractions representing the resources of a physical device, the most important of them being the Domain, the Program Invocation and the Variable abstractions.

TASE.2 [IEC96a][IEC96b][IEC97][KEM96] is a companion standard of MMS. It relies on MMS services and abstractions. For example, TASE.2 defines a Virtual Control Centre (VCC) built with the Virtual Manufacturing Device (VMD) abstraction. The VCC defines a virtual power plant or a control centre. It encapsulates a set of resources like the VMD does. TASE.2 uses other MMS abstractions. As far as we are concerned, in our study we needed the following MMS abstractions: VMDs, named variables, list of named variables and domains.

### 3.2. Client/Server interactions

TASE.2 is Client/Server based, as defined in MMS, and provides its own Object Model. A TASE.2 association is established between two TASE.2 entities. TASE.2 association Objects are mapped on MMS association objects.

TASE.2 objects and services are mapped onto MMS abstractions and services. As a server, a control centre shows its real resources as a subset of available data. Different clients can access these data (one or more other control centres), each client can have different views, and clients can be servers themselves. Two types of interactions are provided. Client initiated interactions are called "operations" and are mapped on classical MMS confirmed services (Conclude and Abort services are invoked by servers but are considered as operations). Server initiated interactions are called "actions" and are mapped to MMS un-confirmed services. Four semantics are provided to exchange data between control centres: "once" (immediate client/server request), "periodic" (periodic transfer), "exception"(state change based transfer), "event" (event condition based transfer).

Access control is handled at the organization level between control centres through a Bilateral Agreement. A bilateral agreement defines formally the set of data that are exchanged between control centres. The real representation of these data is called a Bilateral Table. Each data of the agreement has an entry in this table. It is the responsibility of servers to ensure access control. This feature is insufficiently specified in the TASE.2 standard. We asserted this weakness in the design of TASE.2. Despite the need of access control for EDF, this feature has been eliminated from the study with EDF agreement.

### 3.3. Conformance Blocks

TASE.2 functions are separated in nine conformance blocks: The repartition of functions onto blocks brings modularity to TASE.2, the supplier is free to implement any blocks except the first one that is mandatory.

Block 1 corresponds to basic services. It deals with Association Objects, Data Value Objects, Data Set Objects, and Data Set Transfer Set Objects and some of their condition monitoring. Block 2 deals with more condition monitoring related to Data Set Transfer Set Objects. Block 3 deals with blocked transfers. Block 4 is dedicated to Information Message Objects. Block 5 involves Device Objects control. Block 6 is related to Program Objects. Block 7 contains all functions that handle events. Block 8 deals with accounting. Block 9 deals with time abstractions and full condition monitoring of associated Transfer Set Objects.

Due to EDF's needs, our work has been limited to the evaluation of blocks 1,2 and 5. They are described hereafter.

## 4. Indication Point Management in TASE.2

Conformance blocks 1 and 2 deal with variables called indication points. Indication points can be gathered in complex data structures. Block 1 provides the periodic transfer of power data whilst block 2 implements event- based transfers, block 2 is also referred to as Report-By-Exception (RBE).

Indications points are exchanged using the following TASE.2 basic types of objects that are closely related: Data Value, Data Set, Data Set Transfer Set (DS Transfer Set). Data Values objects references different kinds of indication points. Indication points are status points (bits strings, Data_State, or integers, Data_Discrete), analog points (reals, Data_Real). Indication points can be simple values, with added attributes : quality code (Data_Flags), time stamp (Data_TimeStamp), change of value counter (COV_Counter) [KEM96]. The set of possible indication points are given in the table 1. below.

| Simple types | + quality code | + timestamp | + COV_Counter |
|---|---|---|---|
| Data_State | Data_StateQ | Data_StateQTimeTag | Data_StateExtended |
| Data_Discrete | Data_DiscreteQ | Data_DiscreteQTimeTag | Data_DiscreteExtended |
| Data_Real | Data_RealQ | Data_RealQTimeTag | Data_RealExtended |

**Table 1.** Types of TASE.2 Indication Points

Allowed operations on Data Value objects are: Get Data Value, Set Data Value (the client needs write access), Get Data Value Name, Get Data Value Type. Data Sets are ordered lists of Data Value object identifiers. The most important objects in conformance block 1 and 2 are DS Transfer Sets, they drive the sending process of Transfer Reports from a server to its client. They contain a set of transmission parameters that define under what conditions the data values related to a Data Set have to be transmitted. The number of handled parameters depends of the

conformance blocks supported by the TASE.2 platform. The meaning of the different parameters is given hereafter in Table 2. Figure 2 gives an overview of the Data Object Management framework with Transfer Reports.

| **Attributes** | | | **Meaning** |
|---|---|---|---|
| **EventCodeRequested** | | | Application event specification |
| **DS Transmission Pars** (condition monitoring parameters for the server) | **Interval** (1) | | Time interval between reports |
| | **Start Time** (1) | | Time value to begin the periodic reporting |
| | **RBE, Report By Exception** (2) | | Boolean, false: periodic reporting, true: event base reporting |
| | **Integrity Check** (2) | | Time value defining an interval for integrity check when IntegrityTimeOut condition is used. |
| | **TLE, Time Limit for Execution** (2) | | Deadline for reporting |
| | **Buffer Time** (2) | | Time interval for buffering the ObjectChange condition before reporting. |
| | **Critical** (2) | | The client needs to ack reports |
| | **DS Condition Requested** (status allowing reporting) | **Interval Time Out** (1) | Indicates whether or not the server shall send a report when the Interval time arrives |
| | | **Operator Request** (1) | Indicates whether or not the server shall send a report when an operator at the control centre requests it |
| | | **Object Change** (2) | Indicates whether or not the server shall send a report when any object in a Data Set changes |
| | | **Integrity Time Out** (2) | Indicates whether or not the server shall send a report of the entire Data Set when the Integrity Check time interval expires |
| | | **Other External Event** | Indicates whether or not the server shall send a report when any other external event condition that is not described in the other conditions becomes true |

(1) states that the attribute is handled by block 1 services, RBE should be set to false,

(2) states that the attribute is handled by block 2 services, RBE should be set to true

**Table 2.** Data Set Transfer Set Attributes

## 5. Device Management in TASE.2

The Device Object represents a physical device (generator, transformer, capacitor, transmission circuit, breaker switch…) that could be remotely controlled by a client. The technical implementation how the real device is controlled (libraries, drivers …) and managed is outside the scope of the TASE.2 standard. TASE.2 provides a convenient abstraction to model raw devices.

Two types of objects are defined in the conformance block 5. Devices subject to inter-locking, cannot be accessed concurrently, they need a serialized access. Other devices can be accessed simultaneously. The first ones need to be selected before any operation. Then, the block 5 provides the following operations: Select, Operate, Get Tag and Set

Tag. Select is used for serialized access device objects, if it succeeds the device state goes from IDLE to ARMED. Operate is used to act a device, a select-before-operate device has to be in the ARMED state before being operated. A tag is used to tell if the device can be operated, Get Tag, provides its current value. Set Tag is used to modify its value.

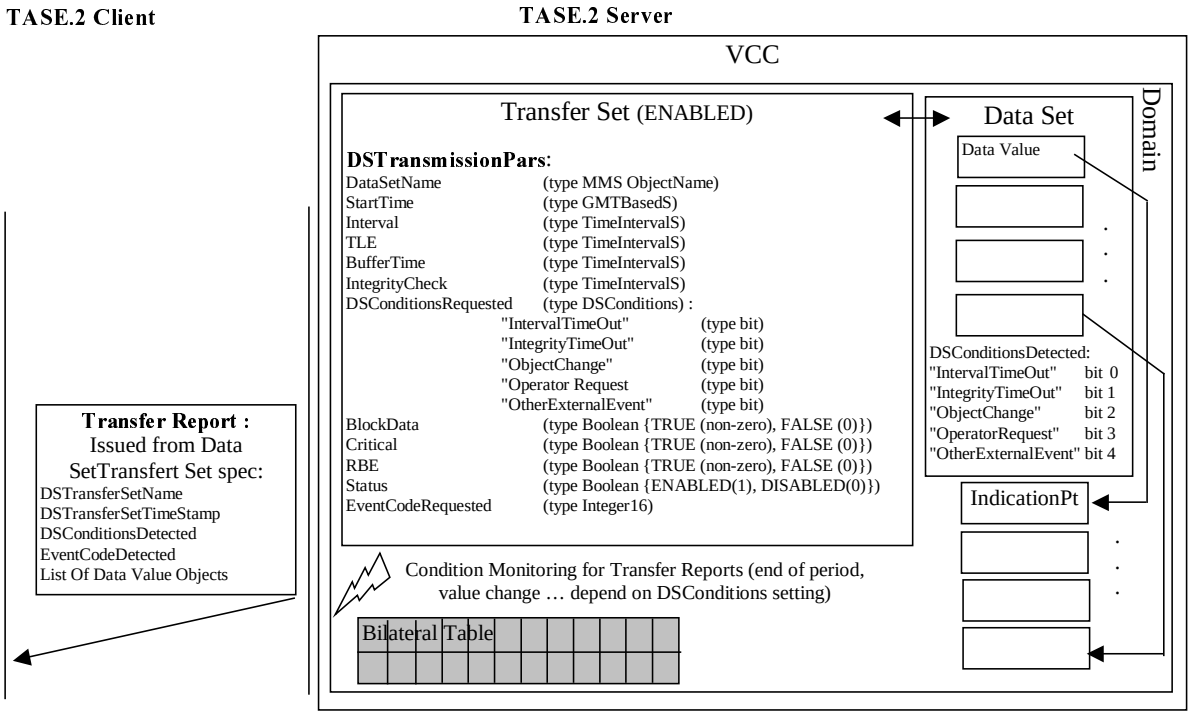


**Figure 2.** Data Object Management framework

## 6. Porting TASE.2 over pSOS+, a real-time kernel

### 6.1 Addressing EDF real-time requirements

The initial goal of our study was to demonstrate the feasibility of a port of a TASE.2 server product over a real-time kernel. The processor that has been used in this experiment, is a MBX board with a Motorola PowerPC 821, clocked to 40 Mhz. It is typical of embedded systems.

Time features included in TASE.2, and power production management real-time constraints, leaded EDF to study TASE.2 over a real-time kernel. The pSOS+ real-time kernel has been chosen by EDF. pSOS+ is supplied by Wind River, (formerly from ISI), it is dedicated to embedded systems. It has been chosen for its properties, the facilities provided by its software development environment. Also, it has been evaluated in a previous study [BEC01a].

pSOS+ is a modular real-time kernel. Its overall architecture is depicted in figure 2. The pSOS+ components are of two types : mandatory and optional software components.

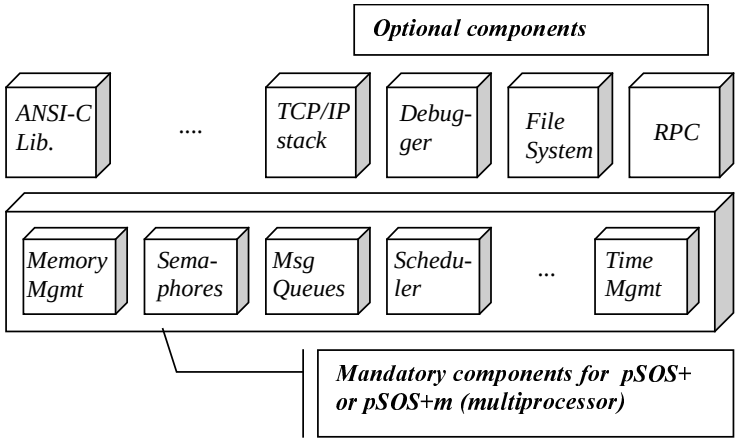


**Figure 2.** pSOS+ real-time kernel architecture

### 6.2. Porting a TASE.2 product

- The first phase deals with porting the server source product code over pSOS+ This work has not been so complex. It took four man months. We got the source product for a Windows NT platform (the reference platform):

  - The first step includes learning of the package, getting its documentation in mind, exercising configuration tools and practice with examples. This step continued with a careful translation of the overall software from one platform to the other. This work was mainly dealing with include files, and makefiles. There is some difference between Windows NT and pSOS+ which is closer to Unix. This phase ended with the port itself. The port of network functions was merely a question of pSOS+ configuration (in terms of number of sockets allowed, buffers…). Only time functions required some code rewriting, it has been done using functions that are almost a standard in real-time systems (the only differences are in function prototypes). A port to another real-time system would be very easy we guess. This step has not been so difficult and took one and a half man months of the all study. This long time is due to the fact that we received three successive versions of the source product during the project. Hotline was provided by GHF automation GmbH, their expertise helped us to learn how the product implemented some specific aspects of the TASE.2 standard.

  - The second step is related to port runtime. In this step, we expected that all the software should run correctly independently of the reference platform. This was difficult because we were completely new on the TASE.2 stuff. This step includes the implementation of an intrusive test tool described further. This phase has been longer than we expected. It took two man months and a half. Main problems came from the rules adopted by the authors of the product source code. For example, the same type of data was defined in different places in the code. One modification in one place had to be reported in other places, and we were

not aware about that through the documentation, we discovered it and fixed this problem. Also, the port required a deep exploration of the source to validate TASE.2 services and abstraction implementations on the new platform.

- The last phase of the port deals with functional testing. It took four man months. The elapsed time in this phase includes documentation. Functional testing adds test scenarios used to stress the port (MBX821-PPC/pSOS+) against the reference platform (Windows NT). This phase has been time consuming. The reason is that we needed a deep investigation to ensure that our port was correct, and that we did not introduce new errors. We had to explain differences of results between the two platforms when they occurred, and to check if it was the platform, and not the port, that provided new behaviors. This needed a feedback to the GHF Automation hotline. This requirement led us to specify tests and test tools to achieve some kind of conformance tests of the TASE.2 server in a context of interoperability. Functional tests used a TASE.2 client from a different supplier. This process will be explained in the next sections.

The overall project took eight man months. The port has been successful at the end. The port over the MBX821-PPC/pSOS+ platform behaves exactly like the TASE.2 product over the reference platform. Also our port fixed some bugs on the reference platform.

## 7. Testing approach and test platform

### 7.1. Test platform

The test platform is depicted in figure 3. A TASE.2 client from another supplier is able to provide operations or receive actions from two TASE.2 servers: the reference server under Windows NT, and the implementation under test. Computers use a dedicated 10Mb/s Ethernet LAN. The MBX/PPC that runs pSOS+ is a target controlled through a PC host under Windows NT. The TASE.2 server over this computer is downloaded from the host, its configuration is hard coded at compilation time.

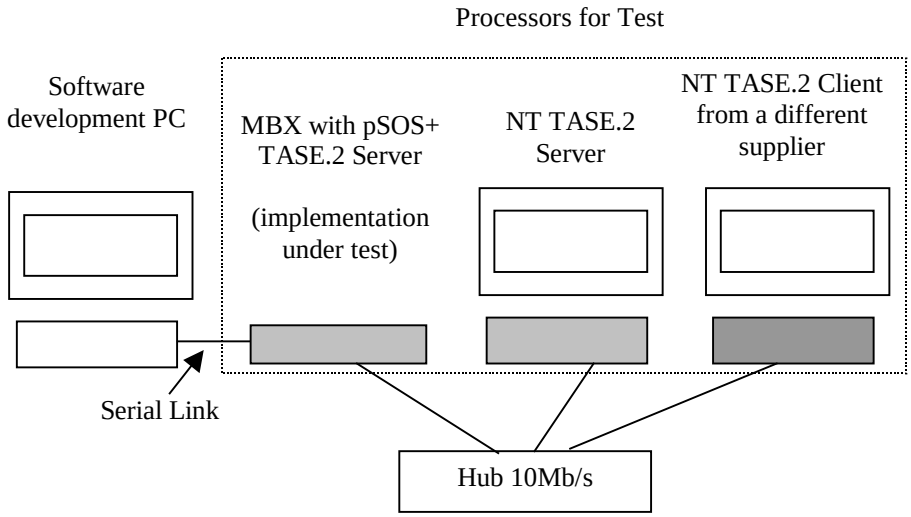


**Figure 3.** Test Platform

### 7.1. Testing

To validate our port, we wanted to test its behavior against the reference platform. The same scenarios are executed from the client towards each of the servers to compare behaviors. Each test scenario is executed, with the TASE.2 server under NT first, the TASE .2 client gets the results. The same test scenario is repeated by the client but with the pSOS+/MBX server. Logs from the TASE.2 client are written for every test and every server.

Tests are positive if logs are the same, except identifications (NT server and pSOS+ server have different IP addresses) and timing (each TASE.2 message is stamped with a date). Traces are generated at the pSOS+/MBX server.

During these tests, we admit that if a test is not working under Windows NT, it is not necessary to test the same function under pSOS+. In fact, we observed in very few cases that same wrong behaviors occurred on both servers.

A full set of scenarios covers conformance block 1, 2 and 5. They ensure that our port acts as the reference platform for every features provided by the TASE.2 standard for these conformance blocks. The specification of these tests has been made carefully. It is mandatory to cover each service for each one of the objects. They took a long time. We called the full test campaign: functional testing.

We estimated at the beginning that a black box approach would be sufficient. The main reasons were that we expected the same behavior between the Windows NT server and the pSOS+ server, independently, from the supporting operating system. We expected that the same source would produce the same exchanges of messages.

This hypothesis was right but required a LAN analyzer. We didn't have such tool. We could have used an analyzer for TCP/IP traffic. It is easy to get one on the web: many free software tools exist. But this approach still needed to

parse the MMS protocol footprint and TASE.2 exchanges. It was too time consuming. This led us to examine more carefully outputs: traces, logs, messages at user interfaces. It became more difficult to recognize which event produces which output, especially in our context when a server is triggered from a TASE.2 client issued by a different supplier. We needed a more intrusive approach to observe the internal behavior of the implementation under test. We needed to be able to observe and to modify the value of variables and their attributes. For these reasons a TASE.2 monitor has been implemented. The conclusion is that we moved to an open box approach, and we realized some kind of conformance testing.

### 7.3. TASE.2 Monitor

The monitor is available on Windows NT and pSOS+. The TASE.2 monitor is a set of introspection functions. Monitor is used to perform “white box” tests which were mandatory to validate in a complete way our port. It accesses almost every internal structure of the server. This monitor was used also to debug the port, but it has been quickly very useful to implement our tests (for example to simulate a physical process that performs a write variable on the server side).

Our TASE.2 monitor offers a wide range of functions, like complete transfer report handling (can change every attribute, RBE, time…), data value write/read… A subset of available functions is given below.

- General information: vendor, model, revision, conformance blocks…
- List of domains.
- Information about a Domain: name, number of variables, number of variable lists…
- Information about a Transfer Set: Range, name, date, periodic interval, TLE, Integrity Check, RBE…
- Modify a Data Set Transfer Set: all attributes can be changed.
- Information about a Device: Name, Domain it belongs to, Class, Type, State, Tag…
- Modify a Device: all attributes can be changed.
- Date: gives current date in TASE.2 format, Unix Format and readable.
- Read a Data: Name, Size, Type and Value.
- Write a Data or Structure: main types can be written using the monitor (integer, float, strings, bit string…)
- Trace on/off: activates or not logs on server side, logs are sent through the network toward the host.

## 8. Functional Testing

Test scenarios cover a wide range of the features provided by conformance blocks 1, 2 and 5. The list of object types stressed by our functional tests encompasses: VCC, association, bilateral table, access control parameters, domain, data value (VCC scope and domain scope), data set, data set transfer set (in fact, data set and data set transfer set tests have been described but the TASE.2 client did not allowed to handle them during execution, they are allocated at the beginning), data set transfer set conditions, transfer reports, select-before-operate devices, concurrent access devices. 50 test cases have been specified for conformance blocks 1 and 2. 12 test cases have been specified for conformance block 5. We execute every possible action for every object type available in TASE.2 blocks we are interested in (blocks 1,2 and 5). In fact, these are almost conformance tests. Table 3 gives the definition of tests for data value objects.

Functional tests have been translated in configuration files for client and servers (text files and C files with C functions to handle special cases like data sharing on server side). These files are based on above tests. Monitor is widely used to execute these tests.

The test campaign provides 104 results related to the previous tests cases. Table 4 summarizes the results for data value objects. The full campaign needs a full day.. A last category of test runs during 2 days. These last ones stress the implementation under load. The TASE.2 monitor has been widely used.

The difficulties we encountered are the following:

- The client and the server came from different supplier, the client did not offer all the flexibility we'd like, we mentioned the fact that data set and data set transfer set were not accessible during execution, so we could not observe the allocation of data set transfer set dynamically.
- The products we tested were not certified: no TASE.2 conformance, nor TASE.2 interoperability. This has a consequence on our test methodology. We adopted the following check-list in case of an un-expected behavior, the order of actions is important :
    - The test specification is wrong ?
    - Did we misunderstand the TASE.2 standard ?
    - Did we make a wrong configuration of the server or the client ?

| **Objects** | **TASE.2 Services/Functions** | **Client side, behavior and expected results** | **Server side, behavior and expected results** |
|---|---|---|---|
| **Data Value**<br>There is as many types of variables as defined in the TASE.2 standard. | **Get Data Value Names** | For each data value, access to its name<br><br>Should be seen in application logs and at the user interface | The server gives data value names.<br>Should be seen in console logs. |
| | **Get Data Value Type** | For each data value, access to its type<br><br>Should be seen in application logs and at the user interface. | The server gives the data value types .<br>Should be seen in console logs. |
| | **Get/Set Data Value** | **For each data value, with read and write access rights, get its value following these services :**<br>▪ Read data values, Get()<br>▪ Modify data values, Set()<br>▪ Read new data values, and attest the occurrence of modifications, Get()<br>Should be seen in application logs and at the user interface. | Data values are all set to an initial value before any access. On the first Get(), this value is provided to the client. The next Set() adds a constant on a data value. The last Get() attests the result of the addition.<br><br>Should be seen in console logs. |
| | **Access Control** (Write Access, Read Access) | Try to write a data value with readonly access<br><br>Should be seen in application logs and at the user interface. | Access denied in any case<br><br>Should be seen in console logs. |
| | | Try to write a data value with readonly access<br><br>Should be seen in application logs and at the user interface. | Access denied in any case<br><br>Should be seen in console logs. |
| | **VCC Data Value** | Define a simple integer data value with read access within a VCC outside a domain, use Get() to read content<br><br>Should be seen in application logs and at the user interface. | The Server provides the initial value contained in the data value.<br><br>Should be seen in console logs. |
| | | Define a simple integer data value with read and write access within a VCC outside a domain, use Get() then Set() and finaly Get(), the new value should be visible by the second Get<br><br>Should be seen in application logs and at the user interface. | The data value is set to an initial value before any access. On the first Get(), this value is provided to the client. The next Set() adds a constant on a data value. The last Get() attests the result of the addition.<br><br>Should be seen in console logs. |

**Table 3.** Test Specification for the TASE.2 Data Value object related to Indication Points

| Objects | Serv/Fonc TASE.2 | TASE.2 Client | TASE.2 Server over Windows NT | TASE.2 Server over pSOS+ on MBX board | Status | Adv |
|---|---|---|---|---|---|---|
| **Data Value** There is as many variables as types defined in the TASE.2 standard. | **Get Data Value Names** | Access the name for each variable<br>Should be seen in application logs and at the user interface. | Server gives every time variables names.<br>Should be seen in application logs and at user interface. | Same behavior | | **F** |
| | **Get Data Value Type** | Access type for each variable.<br>Should be seen in application logs and at the user interface. | Server gives every time variables types.<br>Should be seen in application logs and at the user interface. | Same behavior | | **F** |
| | **Get/Set Data Value** | **For each variable, set to be read/write, access to its value, doing the following :**<br>- Reading variables value,<br>- Modifying variables value,<br>- Reading again variables value,<br>observing change.<br>Should be seen in application logs and at the user interface. | Initial value A=0 (First time server executes).<br>Read the initial value.<br>For a read/write variable, the new value is written with Set and this new value is observed with a Get.<br>Logs validate these operations. | Same behavior | It is necessary to set the transfer report interval to 60s at least to be able to see something on screen.<br><br>Be careful! You have to write a float using: 1,2 and not 1.2, to avoid a type error from OPC client (high level TASE.2 client). | **F** |
| | **Access Rights** (Write Access, Read Access) | Try to write a read-only variable. Should be seen in application logs and at the user interface. | Deny in any case.<br>Should be seen at the user interface. | Same behavior | | **F** |
| | | Try to read a write-only variable. Should be seen in application logs and at the user interface. | Deny in any case.<br>Should be seen at the user interface. | Same behavior | | **F** |
| | **Data Value VCC scope** | Define an integer read-only variable, in the VCC, not in a domain. Access it with a Get() operation.<br>Should be seen in application logs and at the user interface. | Server gives value of the variable.<br>Should be seen at the user interface. | Can be observed very precisely using monitor on pSOS+ server side. | . | **F** |
| | | Define an integer r/w variable, in the VCC. Access it with Get() then Set() it with a new value and finally Get().<br>Should be seen in application logs and at the user interface. | Server returns A, then changes the variable to a new value, then returns the new value.<br>Should be seen at user interface. | Can be observed very precisely using monitor on pSOS+ server side. | | **F** |

**Table 4.** Tests results for Data Value objects related to Indication Points

EB, EGS, EL, GH

- The port is wrong ?
- The reference platform is wrong ?
- The client is wrong ?

We experimented each item of the list except the "client is wrong". We got three times "the reference is wrong". But twice, the TASE.2 standard was insufficiently specified (access control features).

We expect that if we had TASE.2 compliant platforms (conformance and interoperability) our study would have been simpler and shorter. In error cases, we would have suspected our port and only it! Also, the TASE.2 monitor has been very useful for functional testing. But one can argue that this tool is not neutral, because of its intrusion inside the implementation.

## 9. Conclusion

The port has been successful at the end of the project. The port over the MBX821-PPC/pSOS+ platform behaves exactly like the TASE.2 server over the reference platform. Also our port fixed some bugs (very very few) on the reference platform.

We believe that a port over another real-time operating system like VxWorks or LynxOS that we evaluated previously [BEC01a] will be easier, taking into account the experience we gained. We know the source code, we know exactly how it behaves, and functional tests are defined and operational. Also, VxWorks or LynxOS are close to pSOS+, sources will easier to translate. We could think about Real-Time Linux too. We believe that the time for the full project would take half of the time for a new real-time kernel.

One could wonder if using a real-time operating system is mandatory or useful because the reference implementation of our TASE.2 server is running under Windows NT which is not real-time. In our opinion, real-time properties offered by kernels like pSOS+ are not used in this product. There is no control to ensure that deadlines are respected and transfer reports are sent in time. pSOS+ can help to meet such requirements, it allows to run a TASE.2 server with a high priority, meeting real-time constraints in a performance basis (it can be verified using static analysis methods).

Our test suite is close to a conformance test suite. We can assert that conformance and interoperability tests towards the TASE.2 standard are mandatory requirements previous to a port. This is a general assessment, true for any communication protocol [ITC92]. This can shorten and ease the port drastically because when errors occur, you only have to suspect your port and only it! We guess that if we had such tests made before our port, we would have gained two man months.

## References

[BEC01a] E. Becquet, P. Decogné, E. Gressier-Soudan, L. Bacon. "Evaluation de Micro-Noyaux Temps Réel pour les Applications d'Informatique Industrielle d'EDF". In Proceedings of Real Time Systems 2001, RTS'2001. Paris. France. Mars 2001.

[BEC01b] E. Becquet, L. Réveilleau, J-M. Douin, E. Gressier-Soudan, F. Horn, L. Bacon. "Towards a Java Based Embedded Remote Monitoring Tool for Small and Medium Power Plant Units". submitted to JISA 2001. New-York. June 27-28. 2001.

[BAC99] L. Bacon, E. Becquet, E. Gressier-Soudan. "Etude comparative de micro-noyaux temps réel du commerce". 7ème Séminaire du Laboratoire d'Ingénierie de la Sûreté de fonctionnement (LIS). Bordeaux.16-17 Mars 1999.

[GUI00] B. Guillon. "Web based lightweight HMI for Power Plant Control". DEST final report. September 2000. (French)

[IEC96a] IEC, Utility Communications Specification Working Group. "TASE.2 Services and Protocol". Version 1996-08. IEC870-6-503. ICCP Inter-Control Centre Communications Protocol Version 6.1. August 1996.

[IEC96b] IEC, Utility Communications Specification Working Group. "TASE.2 Object Models". Version 1996-08. IEC870-6-802. ICCP Inter-Control Centre Communications Protocol Version 6.1. August 1996.

[IEC97] IEC, Utility Communications Specification Working Group. "TASE.2 Profiles". Version 1996-11. IEC870-6-702. ICCP Inter-Control Centre Communications Protocol Version 6.1. February 1997.

[ITC92] ITCIM Consortium. "Integration Testing for Computer Integrated Manufacturing : Test Methodology". Acerli. Esprit Project N° 6860. Esprit 3. WP200. Deliverable 210.1. December 1 1992.

[KEM96] KEMA-ECC. ICCP User Guide. Final Draft. Mineapolis. October 8 1996.

[LEG00] E. Legrand, E. Becquet, E. Gressier-Soudan, L.Bacon, O. Mallet. "SETR : Systèmes d'Exploitation temps réel : Portage d'une souche TASE.2 sur pSOS+". HP-32/00/057/A. Octobre 2000. French Report, Restricted access.

[MOD01] Modbus/TCP HomePage. http://www.modicon.com/openmbus/. March 22, 2001.

[NIL96] K. Nilsen. "Issues in the Design and Implementation of Real-Time Java". Technical Report. NewMonics Inc. 1996.

[REV01] L. Reveilleau. "Prototypage d'une station légère pour la supervision à distance d'unités de production d'énergie petites ou moyennes". Engineering Degree. Cnam. Paris. March 22th 2001.

[ROS87] M. T. Rose, D.E. Cass."ISO Transport Services on top of the TCP Version 3". RFC 1006. May 1987.

[TAN92] A. Tang, S. Scoggins. "Open Networking with OSI". Prentice Hall. 1992.

[VAL92] Valenzano. Demartini. Ciminiera. "MAP and TOP Communications". Addison Wesley, 1992