Portability Experiment for CTRON Communication Control
(Transport Layer and Session Layer)

Yasuaki Mochida  Hideyuki Matsuda
Fujitsu Limited
Kawasaki 211, Japan

Hiroyuki Ikuta  Toshiya Ohno
Hitachi Limited  NEC Corporation
Yokohama 244, Japan  Abiko 270-11, Japan

Abstract
In the Step2 of the portability experiment conducted by the CTRON Technical Committee, we have ported CTRON specification Extended OS (transport layer and session layer communication control programs) developed by each of the participating companies to CTRON specification Basic OS developed by other firms to run on different hardware. The aim of this experiment is to evaluate the success of CTRON Basic Communication Control specifications. In this paper, first we show the outline of this experiment. Then, we discuss impediments with porting CTRON Basic Communication Control products and evaluates the scale of modified lines and the number of man-hours required for the porting process. Finally, we suggest how to solve the problems on porting programs.

1. Introduction
The CTRON interface specifications, developed by the CTRON Technical Committee of the TRON Association, have as one aim improving software portability. The CTRON interface specifications[1][8] have already been opened around the world, and a number of operating systems have been developed based on these specifications. The portability experiment was divided into two steps in order to evaluate the portability of CTRON specification Extended OS. Step1 began in June 1990 and was completed in May 1991. Step2 began in September 1991 and was completed May 1992[9]. Extended OS and APIs were ported to the basic systems consisting of hardware and Basic OS, installed at the University of Tokyo. In Step1, Extended OS programs, not including communication control, were ported to each basic system individually[10]-[15]. Then in Step2, Extended OS programs including those with communication control functions were ported, and the various basic systems were interconnected to confirm their interpretability.

In this paper, we describe three portability experiments in Step2 that transport layer communication control functions and session layer communication control functions have been ported.

Section 2 presents the aims of this experiment. Section 3 outlines three portability experiments and points out the differences between source system and target system, and each experiments features, furthermore the solutions to the problem areas considered prior to the experiment. Section 4 presents a detail analysis of the experimental results by classifying and evaluating the problems at the each impediments. Section 5 evaluates portability of CTRON Basic Communication Control, and finally shows how the results are being reflected in CTRON Technical Committee activities.

2. Aims of the Experiment
The objects of CTRON specifications are OS interfaces. The internal details of the OS are not specified, nor is the method of implementation. In drawing up CTRON specifications there were certain matters that could not be standardized and had to be made implementation-dependent, since they are strongly influenced by service conditions and system configuration. There are also some matters that are not included within the scope of CTRON specifications but are in fact a necessary part of OS design, such as system startup and fault processing. In addition, there is always the possibility that interpretations of the specifications will differ from one implementor to another. And development environments are outside the scope of the interface specifications.

The aims of the experiment is therefore to verify the degree to which software developed based on CTRON specification is able to be ported, furthermore...
to confirm communication between systems that
communication control functions and application
programs were ported.

3. Outline of Experiment

3.1 Porting process

The porting process can be classified into four
steps, as follows.

- Adjusting the development and test environments
- Determining the places to be changes
- Program change
- Test

In preparing for the experiment Step 2, in order to
allow porting to take place smoothly, we considered
the problems as pointed out in section 3.2.

3.2 Problems areas considered prior to
the experiment

In drawing up the Basic OS interface specification,
it was necessary to deal with the technical con-
fusion between assuring realtime performance and
hiding differences in hardware. Some of the inter-
faces are therefore specified only loosely, leaving
details to the implementor. There are a number of
implementation-dependent aspects for which each
firm may have a different solution. Even where in-
terfaces are specified in detail, there are also in-
stances where implementations differ due to ambi-
guities in the specification or interpretation. In
addition, there are differences in development
environments, such as OS, compiler, and software tools.

In actually porting the software, there will need
to be certain adjustments to resolve the problems
caused by the above originating points of im-
pediments. The main problems are classified, as
follows.

(1) Main CTRON communication control specifi-
cation-related issues

- Communication control buffer management
  Communication control buffer management is de-
ned as a layer-common function, but there is no
  specifications as to which layer is to provide this
  buffer management.
- Relation to device management
  Relation to Im interface, circuit management, and
  CCL initialization procedures are not specified.
- SAP address format
  Version 2 of the CTRON communication control in-
terface specifications indicate an addressing scheme
in the common items, but in the language binding
specifications, the C language address type defi-
nition for each layer is simply given as (char*).
Implementors of the specifications to date are di-
vided into those that follow the OSI communication
control processing method in stacking each
SAP(service access point) address as in INAP, and
those that adopt a character string method that
identifies a single layer.
  - Buffer pool information area type
    The type of calling buffer pool information ac-
quired by CM-GET-BPI is not specified.
(2) Problems relating to differences in the basic
systems specifications

- Calling task information area type
  The type of calling task information acquired by
GET-TSI is not specified.
  - Size of stack area
    One basic system implements a function for auto-
matically extending the stack area. In Another basic
system, it is necessary to designate the stack area
size required for a task in the user work parameter
with CAS-TSK.
- Assignment task execution levels and priorities
  The range of these values are dependent on Kernel
implementation and the assigned values should be
decided by system managers.
  - Size of int-t type
    Integer type specified in CTRON general rules for
C language binding is named int-t [1]. For example
size of int-t type is 32 bits or 16 bits, it de-
pends on CPU architecture.
- Extended OS system call registration method
  The method for Extended OS system call regis-
tration is not specified.
  - System initialization
    For initialization, the specifications for each in-
terface class stipulate only system call names.
    Functions and parameters are left to the implementor.
  - SG data of Extended OS
    It is not specified how to dispose the SG data of
Extended OS.
  - Problems relating to development environment dif-
ferences
    Development environments are outside of the in-
terface specifications, but there are important in
achieving portability.

The above instances are common problems in CTRON.
These problems have already revealed in Step 1 of
the portability experiment. The problems are clla-
ssified to the following 4 parts by the porta-
bility impediments.
(3) Problems relating to communication test

In communication control, specifications are given for each OSI layer separately, and portability across different systems is essentially possible so long as the layer match.

The portability impediments and solutions are summarized in Table 1.

Table 1. Main portability impediments and solutions

<table>
<thead>
<tr>
<th>Problem</th>
<th>solutions (Experimental spec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication control buffer management</td>
<td>Not specified</td>
</tr>
<tr>
<td>SAP address format</td>
<td>Implementation-dependent</td>
</tr>
<tr>
<td>Relation to device management</td>
<td>Not specified</td>
</tr>
<tr>
<td>Buffer pool information area type</td>
<td>Not specified</td>
</tr>
<tr>
<td>Calling task information area type</td>
<td>Not specified</td>
</tr>
<tr>
<td>Size of stack area</td>
<td>Implementation-dependent</td>
</tr>
<tr>
<td>Assignment task execution levels and priorities</td>
<td>Implementation-dependent</td>
</tr>
<tr>
<td>Size of int_t type</td>
<td>Implementation-dependent</td>
</tr>
<tr>
<td>Extended OS system call registration method</td>
<td>Not specified</td>
</tr>
<tr>
<td>System initialization</td>
<td>Not specified</td>
</tr>
<tr>
<td>SG data of Extended OS</td>
<td>Not specified</td>
</tr>
<tr>
<td>differences in development environments</td>
<td>Development environment</td>
</tr>
</tbody>
</table>

CTRON is premised on porting of source code as a rule for differences in the compilers.

Table 2. Main problems and solutions in communication test

<table>
<thead>
<tr>
<th>Layer</th>
<th>problem</th>
<th>solutions (Experimental spec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session layer</td>
<td>Protocol version</td>
<td>Version 1 is to be used</td>
</tr>
<tr>
<td></td>
<td>Function units</td>
<td>Minimum kernel subset and full duplex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are used</td>
</tr>
<tr>
<td></td>
<td>Segmentation</td>
<td>Not being used</td>
</tr>
<tr>
<td>Transport layer</td>
<td>Protocol class</td>
<td>Class 0 is to be used</td>
</tr>
<tr>
<td></td>
<td>Protocol class negotiation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TPDU length</td>
<td>256 is adopted</td>
</tr>
</tbody>
</table>
In this experiment each of the communication control programs were interconnected after porting. To this end it is necessary to confirm the interpretability of each of these programs. This is being done by having each firm fill out the PICS (profile implementation conformance statement) provided for the INTAP implementation conventions, and comparing these with each other. Table 2 shows the main problems and solutions relating to confirming interpretability.

### 3.3 Configuration of the experiment

Two participating companies were providing basic systems (target machines for the experiment). These machines run a CTRON-specification Extended OS providing layer 3 interface functions (X.25). Each of the three firms ported to these two machines a CTRON-specification Extended OS providing a layer 5 interface with layer 4-5 (transport and session layer) functions. Two target machines were interconnected via an ISDN packet network and evaluated. The experiment for this part of the evaluation is configured as shown in Figure 1.

![Configuration of the experiment](image)

**Table 3. Differences in experiment conditions**

<table>
<thead>
<tr>
<th>Item</th>
<th>[A]</th>
<th>[B]</th>
<th>[C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware architecture</td>
<td>not equal 32bits to 16bits</td>
<td>not equal 16bits to 32bits</td>
<td>equal 16bits to 16bits</td>
</tr>
<tr>
<td>Development environment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C language specification</td>
<td>equal ANSI to ANSI</td>
<td>not equal K&amp;R to ANSI</td>
<td>not equal K&amp;R to ANSI</td>
</tr>
<tr>
<td>Compiler restriction</td>
<td>not equal tight to loose</td>
<td>not equal loose to tight</td>
<td>not equal tight to loose</td>
</tr>
<tr>
<td>ex. Translation limits</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Extended OS

Basic OS

Basic system 1

Basic system 2

Fig. 2. Experiment [A]

Fig. 3. Experiment [B]

Fig. 4. Experiment [C]

Kernel subset C: Common part
Kernel subset M: Optional part for task management and task synchronization and communication
Kernel subset I: Memory management optional part
3.4 The differences of the three portability experiments

Three portability experiments are called [A],[B],[C], respectively. The differences between original systems and target systems are shown in Figure 2, Figure 3, and Figure 4. The difference among the three experiments occurs at the conditions as showed in Table 3.

4. Analysis of Experimental Results

4.1 Impediments revealed by the actual experiment

The ratio of changed lines owing to the impediments revealed by the actual experiment at the total changed lines is about 10 percent. (See Figure 5). For example, the impediments relating to CTRON communication control specifications of these were as follows.

- General entry function can not call layer 3 sys-

- tem calls. This function was adopted in the CTRON specification only recently.

4.2 Changes owing to impediment

The scale of changes in these experiments are shown by the ratio of changed line at original size in Figure 6. We have to consider that each experiment conditions is not equal, as noted in section 3 (Table 3).
Determining programs to be changed
Environment adjustment
Program change
Test

Fig.8. Man-hours in porting process

Figure 7 shows the ratio of changes to each of the portability impediments. The data shown in Figure 6 and Figure 7 are analyzed, as follows.

1. The change ratio of source programs in the experiment [B] is much larger than that of the experiment [A] and [C]. (See Figure 6) The reasons are as follows.
   - In the ratio of changes to impediments, the ratio of "development environment (compiler differences)" is 52 percent at the experiment [B]. (See Figure 7) The experiment [B] was seriously affected by compiler differences in the upper limits. The compiler running on the target system have very small upper limits on the number of type definitions, external identifiers, and macro identifiers, etc. For example, the upper limits of the external identifiers number is about one fourth.
   - There were many change places caused by "not specified (Extended OS system call registration method)" in comparison with the experiment [A] and [C]. The change ratio of source programs in the experiment [B] will be almost same as that of the experiment [A] and [C], except the above changes.

As mentioned above, these problems caused by the compiler differences will always arise in porting; some guidance therefore should be referred in a coding manual.

2. The ratio of "implementor-dependent" in the experiment [A] (figure 7) is larger than that of the experiment [B] and [C]. The reasons are as follows.
   - In the experiment [A], size of "integer" in source system is 32 bits and in target system 16 bits. (See Table 3)
   - On the other hand, in the experiment [B], size of "integer" in source system is 16 bits and in target system 32 bits. In the experiment [C], size of "integer" in source system is 16 bits and in target system 16 bits.

Though the differences in size of "integer" depend on CPU architecture, size of int... should be specified as far as the free selection of CPU is guaranteed.

3. There is "unclear specifications" only in the experiment [A] This is the problem of CM... R E B U F as noted in section 3.

4. There are no changes caused by "development environment" in the experiment [A] and [C]. The reasons are as follows.
   - Though compilers are different, there are no changes caused by the upper limits. Because the upper limits are large. (See Table 3)

4.3 Man-hours for each porting steps

The porting process can be classified into four steps (adjusting the development environments, determining the places to be changed, program change, and test). The man-hours required for each of these steps are as shown in Figure 8. The data shown in Figure 8 are analyzed, as follows.

1. The ratio of "test" is large in the experiment [A] and experiment [B].
   - It shows that a key point in accomplishing porting smoothly is how to test efficiently.

2. The amount of "adjusting the development environments", "determining the places to be changed", and "program change" in the experiment [A] is larger than that of the experiment [A] and [C]. The reasons are as follows.

As the promoting company of the experiment [A] were providing a basic system, this company had porting know-how.

The feature of this experiment is confirming the interconnection of each basic system. We anticipated the man-hours required for test steps are large, but actually the man-hours required for this step was
almost same as those in Step1 of the portability experiment. Because CTRON communication control interface specifications are based on OSI service primitives.

5. Evaluation of Portability

Based on the result of these three experiments, it should be possible to port CTRON specification communication control by making changes averaging only 1.1 percent of the original program size. (See the expected value in Fig. 6) This assumes that the problems in the experiments have been fed back to CTRON specifications or portability guide book. From the viewpoint of the number of impediments gained through the experiments, 39 impediments (total 52) will be fed back to CTRON specifications or portability guide book. The main impediments occupied in 1.1 percent are summarized as follows.

- The registration for extended OS system call and control passing method
- Extended OS initialization method
- SG interface between extended OS and basic OS
- Task execution levels and priorities

These problems have already been revealed by the experiment Step1. These interfaces are required for CTRON, we hope to add to CTRON interface specification quickly.

As mentioned above, we could verify that there were no great impediments inherent in CTRON communication control specifications. After this experiment, we have to verify the performance of CTRON specification communication control.

6. In Conclusion

The above overview of the portability evaluation for CTRON communication control has presented portability impediments and solutions and evaluated the degree of CTRON communication control interface in achieving their aim of improved portability. Furthermore, we have confirmed the interconnectability of each resulting system.

The problems in the way of portability that have been revealed by the experiment are being resolved by means of revisions to original specifications and by design guidelines and by coding guidelines, based on feedback from the experiment.

We would like to expect that this experiment will promote OS interface standadization and establish an infrastructure for software portability, as well as leading to multi-vendor software procurement and the further spread of CTRON. And we hope that CTRON activities contribute to improve software productivity.

In conclusion, the authors would like to express their deep thanks to Dr. Ken Sakamura of the University of Tokyo, to CTRON Technical Committee Chairman Dr. Fukuya Ishino, to CTRON Portability Evaluation Working Group leader, and to the corporations in charge of carrying out the experiment.

References