At production sites, secondary tasks such as energy saving, safety management and production streamlining are becoming vital in addition to the primary task of production activities. However, it is difficult to meet these requirements only by changing the functions of field devices. We therefore propose Field Overlay Architecture (FOA) which provides existing devices with a common platform and enables the functions required to handle these tasks to be added via the network at any time. This paper describes how system resources in devices are managed as an important issue for achieving FOA.

**INTRODUCTION**

At production sites, it is required to realize production systems with various additional tasks such as low environmental burden, traceability of product quality, and optimization of the management of equipment and facilities in addition to their primary task of production activities. This trend is accelerating in the 21st century. However, it is difficult for sensors, control devices and actuators (hereafter called “field devices”), which are geographically dispersed at production sites, to meet such a broad range of requirements by modifying them on-site. As a result, the requirements are often met by installing new devices alongside the existing system, or in the worst case the new requirements must remain unsatisfied.

We have proposed a flexible and expandable mechanism for such additional tasks which we call the Field Overlay Architecture (FOA). In this architecture, field devices with a common platform provided in advance are connected through a network and necessary functions are deployed via the network whenever needed. Under the FOA, the field devices are connected to a network, and they can access various information of other devices, and can also perform autonomous and cooperative data processing over the network whenever required.

A major difficulty in implementing the FOA is how to allocate resources to execute additional tasks while avoiding adverse effects on the field devices performing the primary task. Resources in this context refer to computing resources, network resources and field information resources, which we call “field system resources” as a whole.

Traditionally, in order to perform the primary task and expected additional tasks, a system is designed in advance to accommodate these tasks, and the required field system resources are allocated to the devices. In contrast, in the FOA, field system resources are pooled in the devices in advance, and they are allocated dynamically to newly-generated additional tasks after the system is launched.

In this paper, we describe issues relating to the management of field system resources, possible solutions, and examples implemented in the field devices.

**ISSUES CONCERNING FIELD SYSTEM RESOURCES**

Devices placed throughout production sites are required to be compact due to space constraints and other factors, and to consume little power due to power supply issues caused by the installation environment such as the explosion-proof requirement. They should also be reasonably priced. To meet these requirements, the field system resources embedded in field devices are thus restricted and are less than those found in conventional office computers.

This section discusses issues the management of field system resources faces when performing diverse additional tasks on such devices.

**Issues concerning the primary task**

The top priority in the management of field system resources is to maintain field devices’ capabilities to execute the primary task for narrowly-defined production activities safely and without delay. These capabilities include real-
time operation, responsiveness, reliability and availability required for the production activities. Traditionally, the target capabilities are defined when planning a production system, a system that meets the target is designed, and then the field devices are placed accordingly.

Therefore, every time the allocation of field system resources is changed in order to add a new task after the system operation starts, it is necessary to check that all the field devices have the capability to handle their primary task.

**Issues concerning additional tasks**

The execution of multiple additional tasks requires balancing and control to prevent competition for resources or inconsistency of information between additional task systems. In a structure where the field devices and field network allocated to each task are physically separated, it is difficult to execute multiple additional tasks in parallel that require field system resources scattered among multiple devices. Accordingly, the management mechanism must never cause problems even when multiple tasks use the same field device or the network concurrently. In addition, a usable range of the field system resources should be strictly limited by every additional task to prevent each additional task from adversely affecting other additional tasks by its own defects or operating errors.

**Traditional approach**

Traditionally, there are three typical approaches to handling additional tasks. The first approach is to build a mechanism for scanning and processing field information into devices by estimating in advance the additional tasks that might be required. This is suitable for formulated tasks but not for unexpected tasks. The second approach is to expand or modify an existing system to cover additional tasks. This assumes that the field system resources of the existing system are leveraged, which entails risks such as system shutdown during modification, functional degradation of the primary task or complex system implementation. The third approach is to construct a new additional task system independent of the existing system by preparing new field system resources by adding devices and installing the network. Although this is the most suitable system for executing an additional task, it is expensive and unrealistic except for a large task with major cost-benefit performance.

**FOA APPROACH**

We propose an overlay model as the FOA which adds a system to execute newly added tasks while handling the issues of independence between the primary task and the additional tasks and the execution of multiple additional tasks, the issues mentioned in the “Issues concerning the primary task” and “Issues concerning additional tasks” sections respectively. As shown in Figure 1, the person in charge of the additional task can manage and operate his own field system resources allocated to each field device without having to consider the physical structure of the field devices.

![Figure 1 FOA approach](image)

In the additional task system, necessary programs are placed at multiple field devices. These execute the additional task using the system resources provided by the relevant devices. Therefore, in the FOA, the resource management that appropriately allocates the required system resources provided by the field devices to the execution environment created for each task or device, where the additional task programs are executed, is the crucial issue. We have hence proposed a resource management mechanism. (3)

Figure 2 illustrates a model of a single field device under the FOA. FOA Platform in this figure is the execution environment management modules for the FOA. As mentioned in the “Introduction” section, the field system resources are classified into three types: computing resources in charge of program execution, network resources used to communicate with other field devices, and field information resources. Field information resources include information provided by control functions, which is a part of the primary task of the field device, as well as information that can be collected through various I/Os of the devices.

![Figure 2 Field device model under the FOA](image)

In Figure 2, computing resources and network resources are shared by the primary task and the additional tasks. Although the FOA allows separate configuration with multiple hardware resources when strict independence between these tasks is required, this paper focuses on the configuration sharing single hardware.
Resource allocation to execution environment

The resource allocation control for each resource type is as follows.

a) Computing resources allocation control

Computing resources such as CPU processing power or memory are constantly allocated to the primary task as required to ensure its execution, then the remaining resources are allocated to the execution environment for each additional task. In order to utilize the limited resources efficiently, as soon as some additional task is no longer necessary, the resources allocated to that task are deallocated along with its execution environment.

The CPU resource allocation is based on fixed priority control provided by the OS, and the control of CPU usage rate peculiar to the FOA is added to it. In this CPU usage rate control, a CPU usage rate, the ratio of the maximum execution time to control cycle time defined by the FOA, is given to each additional task, and the execution of additional tasks is controlled not to exceed the CPU usage rates on a time-sharing basis. With only fixed priority control, once an additional task with higher priority starts, it continues to use CPU resources until it terminates. In contrast, with the CPU usage rate control added, a task releases CPU resources in accordance with the predefined usage rate, and tasks with lower priority are processed in parallel.

b) Network resources allocation control

The FOA provides a logically-independent network structure to the both primary and additional tasks and enables them to manage and control necessary network resources independently. The FOA assigns a network identifier specific to the task to the execution environment of each field device forming the identical additional task system. It treats a single physical network node as logically-independent multiple network nodes. Subsequently, it allocates the network resources to each execution environment.

The FOA, as with the case of CPU resources, gives higher priority to the network communications of the primary task in order to secure sufficient communication bandwidth. It prohibits additional tasks from using more network resources than necessary by setting an upper limit on the communication bandwidth.

c) Field information resources access control

Multiple additional tasks access the access control function of the field information resources in the execution environment management. The FOA provides a common interface to information resources, controls serialization for exclusive control between additional tasks, and also controls data access privileges based on the role and application of the additional task.

Overlay-type control in execution environment

The FOA offers not only the control of each field device but also the control of each additional task, coordinating across multiple field devices; we call this scheme overlay-type control. This scheme handles all the execution environments of the field devices forming the additional task system as one group and enables the policy given to each additional task to be reflected in the execution environments in the group and the associated field system resources simultaneously.

For example, when the importance of a certain additional task is set to “high,” all the usage priorities of the CPU resources and the network resources in the execution environments in the additional task system will be assigned as “high.”

IMPLEMENTATION OF THE FOA

On an experimental device assumed to be a field device, we built a prototype of the execution environment management modules to realize the FOA as described in the “FOA approach” section. Figure 3 shows the structure of the experimental device. We used Linux as the OS and JavaVM (4), which does not depend on the CPU or OS, as the base of the execution environment for each additional task.

In this section, we clarify issues concerning the resource management provided by the FOA utilizing the resource management of Linux and JavaVM as the base. We also describe the execution environment management modules to resolve these issues.

Figure 3 Structure of the experimental device

a) Allocation of CPU resources

We used the fixed priority control function provided by Linux for allocating CPU resources, and incorporated the accounting system of CPU resources (5) into Linux for CPU usage rate control.

The execution scheduling for additional tasks is controlled by the fixed priority and the CPU usage rate during the remaining time not used by the primary task. Figure 4 illustrates the mechanism. The FOA provides three types of execution control scheme corresponding to the characteristics of additional tasks as follows.

1) The first scheme is for high-speed processing or an exceptional additional task. When the execution is requested, it is scheduled prior to other tasks using the remaining time not used by the primary task. Possible effects on other additional tasks must be considered.
2) The second scheme is for an additional task that requires
a certain volume of processing periodically. This scheme schedules the additional task in order based on its fixed priority and allocates CPU resources to it according to its CPU usage rate for each control cycle within the time that is not used by the primary task and the additional tasks with the fixed priority “high” (additional task 1 in Figure 4). Because CPU resources are allocated on the basis of every control cycle, if the full execution time for the task can not be used because of long-time processing of other prioritized tasks in a control cycle, the unused time is not carried over to the next cycle.

3) The third scheme is for an additional task whose priority is lower than that of the tasks mentioned in 1) and 2). The task is scheduled when CPU resources are available even after resources are allocated to the tasks in 1) and 2). An additional task assigned to this scheme is executed as a background job only when CPU resources not used by other tasks are available.

![Figure 4 Example of CPU resource scheduling](image)

b) Allocation of memory resources

After securing sufficient memory capacity for the primary task, the FOA controls memory resources for every additional task according to its required amount. Although JavaVM can limit the heap area used by an application program directly, it cannot restrict the memory capacity in the OS used when OS functions are called. Thus we added a mechanism in JavaVM to set a limitation on the use of OS functions such as the number of sockets or threads.

c) Allocation of network resources

In order to separate the network structure by every JavaVM, we added a mechanism to JavaVM to assign an IP address to each JavaVM, and assigned a different IP address to each JavaVM. We also allocated network resources to the primary task and each JavaVM utilizing QoS (Quality of Service) control provided by Linux. The experiment proved that the periodic transmission was maintained even during burst transmission.

**CONCLUSION**

As a mechanism to handle a variety of production-related additional tasks, we developed the concept of the FOA, proposed a mechanism to control field system resources, and showed examples of implementation utilizing Linux and JavaVM.

We will investigate migrating the resource management scheme to various field devices and develop a system constructing framework capable of adding additional task programs to the FOA dynamically. We will also conduct experiments to evaluate the functions and performances of the FOA Platform and improve them further.

**REFERENCES**