Abstract—Software component techniques have been widely used for enhancement and the cost reduction of software development. We herein introduce a component system with a real-time operating system (RTOS). A case study of a two-wheeled inverted pendulum balancing robot with the component system is presented. The component system can deal with RTOS resources, such as tasks and semaphores, as components. Moreover, a trace functionality which is a new functionality to confirm the state of components or calling components without modification of C source code is introduced.

Keywords—component-based development; real-time operating system;

I. INTRODUCTION

Recently, software component technologies for embedded systems have received increasing attentions from researchers [1]. For example, component technologies for embedded systems, such as Koala component [2], SaveCCT [3], and THINK framework [4] have been developed.

TOPPERS Embedded Component System (TECS) [5] has been investigated in the last several years. It is possible to estimate the memory consumption of an entire application because TECS adapts a static configuration. The static configuration means that both the configuration of component behaviors and interconnections between components are static. Furthermore, TECS supports optimization of the interface code generated by an interface generator [6]. Therefore, it is easy to understand the software structure because RTOS resources can be represented on a component diagram.

In this paper, through a case study of a two-wheeled inverted pendulum balancing robot, the way to effectively use RTOS resources on our component system and a trace functionality which is a new functionality to confirm the state of components or calling components without modification of C source code is introduced.

II. RTOS RESOURCE AS COMPONENTS

This section describes the overview of TECS and the way to effectively use RTOS resources in TECS.

A. TECS

A cell is an instance of a component in TECS. Cells are properly connected in order to develop an appropriate application. A cell has entry port and call port interfaces. The entry port is an interface to provide services (functions) to other cells. The service of the entry port is called the entry function. The entry port and the call port have signatures (sets of services). A signature is the definition of interfaces in a cell. A cell type is the definition of a cell, such as the Class of an object-oriented language. A cell is an entity of a cell type.

Figure 1 shows an example of a component diagram. Each rectangle represents a cell. The dual rectangle depicts an active cell that is the entry point of a program such as a task and an interrupt handler. The left cell is a TaskA cell, and the right cell is a B cell. Here, each of tTask and tB represents the cell type name. The triangle in the B cell depicts an entry port. The connection of the entry port in the cells describes a call port.
The description of a component in TECS can be divided into three descriptions: a signature description, a cell type description, and a build description. Each description is explained in [5] in detail. An interface generator generates interface code (.h or .c) of the C language and system configuration files (.cfg) for an RTOS from the signature, cell type, and build descriptions.

B. Factory Statement

A factory statement is used to automatically generate a static API for a kernel configuration when cells are instantiated. Figure 2 describes an example of a cell type description to define components for the task of an RTOS.

Figure 3 represents the static API from the factory statement (Lines 12-16 in Figure 2). The static API creates a task with an ID number specified by a task id, such as TASKID_tTask_TaskA, based on the information contained in a parameter of the static API.

C. Composite Cell Type

TECS provides several levels of composition for application developers. A composite cell type includes two or more cells. Figure 4 is one of the examples of a composite cell type. In this case, a periodic task cell type includes three cells: a cyclic handler cell, a cyclic handler body cell, and a task cell.

The cyclic handler cell type (tCyclicHandler) is a time event handler periodically activated to invoke the cyclic handler body. Figure 5 represents the signature description of tCyclicHandler. The siHandlerBody signature is to invoke the cyclic handler body. The sCyclic signature is to manage the cyclic handler cell. Figure 6 represents the cell type description of tCyclicHandler. The tCyclicHandler cell type has two interfaces (Lines 3-4). The factory statement (Lines 11-14) for tCyclicHandler is defined to generate the static API of the cyclic handler.

The cyclic handler body cell type (tTaskActivator) is to activate the task cell. Figure 7 represents the cell type description of tTaskActivator. The tTaskActivator has two interfaces. One is entry port (eiBody) which is invoked
by the cyclic handler and the other is call port (ciTask) which is to activate a task. Figure 8 shows entry function of tTaskActivator. A call port function (Line 6) is used to invoke the function of connected cell. The call port function is generated by the interface generator.

Figure 9 shows the cell type description of a composite cell type. As mentioned above, the tPeriodicTask includes three cells: Task cell (Lines 14-19), CyclicMain cell (Lines 20-22), and CyclicHandler cell (Lines 23-28). The attributes of composite cell type can be specified in the similar way with shown in Lines 7-13. These attributes need to delegate to the attribute of inner cells (Lines 16-17, and 25-27). As well, call port (Line 18) and entry port (Lines 29-31) need to delegate to the interfaces of inner cells. As shown in Figure 10, the way to instantiate a composite cell type is the same as the normal cell type. Therefore, application developers can describe the build description without regard to the composite cell type.

The benefits associated with using a composite cell type are significant reductions in the complexity of component connection because several cells can be treated as a single cell type, which enhances usability through the connection of cells for an application developer (component user). Several component models, such as Koala component and SaveComp component model, also support a component composition mechanism for naming a collection of components and hiding its internal structure. It is, however, more effective to use the composite cell type with factory statements in TECS. For example, a serial port component, one of the composite cell type, includes two factory statements to create semaphores. These semaphores are to manage a send buffer and a receive buffer, respectively. Two static APIs are automatically generated when application developers use the serial port component.

III. TWO-WHEELED INVERTED PENDULUM BALANCING ROBOT

A. Component Diagram of the Robot

Figure 11 shows pictures of a two-wheeled inverted pendulum balancing robot, and Figure 12 represents its component diagram. The robot has three types of sensor: a gyroscope, a rotary encoder, and a current sensor. A gyroscope cell measures an orientation based on the principles of the angular momentum. A rotary encoder cell converts the angular position of an axle to an analog code. A motor current sensor cell detects the electrical current. A/D converter cell converts continuous signals to discrete digital numbers. A motor cell controls a motor. A motor manager cell is activated by a timer to manage the motor. A user control cell performs forward movement, back movement, right turn, left turn, and user commands, from the dataqueue. A ZigBee receiver cell sends the received user command to
the dataqueue. A calc cell is activated by a periodic task to calculate the value for a proportional-integral controller which is a generic control loop feedback mechanism.

As mentioned above, a periodic task cell composes three cells: a cyclic handler cell, a cell to activate a task, and a task cell as shown in the lower left of Figure 12. The same number on the top left of rectangles in Figure 12 represents the same cell type. It is easy to reuse components because the component granularity of TECS is small compared with those of existing component systems.

B. RTOS Resources

When RTOS resource cells, tasks, a periodic task, a dataqueue, and ISR (Interrupt Service Routine) cells are instantiated, the kernel configuration file including the static API is automatically generated. Filled the rectangles in Figure 12 show RTOS resource cells. It is possible to change the configuration files depended on RTOSes, such as µITRON [7] and OSEK [8], by using the factory statement. Therefore, RTOS resource cells are independent of RTOSes.

C. Trace Functionality

A trace functionality supports that the developer can confirm the state of cells or calling cells without modification of C source code. Figure 13 depicts results of software prove for the value of the right rotary encoder.

To realize the trace functionality, a through keyword is used for inserting a cell between cells. The way of using through keyword is explained in [9]. Figure 14 is an example of using the through keyword. The upper part of Figure 14 represents the component models, while the lower part
shows each build description. The through keyword is described only before the description (cCallPort=B.eEntryPort) of the connection. “TracePlugin” represents the type of plug-in. The plug-in is used to define the inserted cells and to generate C-language source code when interface code is generated by the interface generator. The application developers need to select the plug-in provided by component developers for inserting the cell. In this case, the plug-in of TracePlugin is used to obtain the value of the right rotary encoder.

IV. RELATE WORKS

Koala component model [2] is intended for consumer electronics such as televisions. RTOS resources as components, however, are not supported.

SaveComp Component Technology (SaveCCT) [3] is intended for embedded control applications in vehicular systems. SaveCCT supports resource-efficiency, predictability and safety in the vehicular domain. It is, however, easy to adapt other domain because this technology specializes vehicular system. SaveCCT provides a task allocation functionality. Other RTOS resources, however, are not supported.

nesC [10] is to support component-oriented application design. nesC has been used to implement TinyOS [11], a small operating system for sensor networks. The system is generally smaller and provides less functionalities than RTOSes.

THINK framework [4] is an implementation of FRACTAL component model [12]. THINK can also deal with RTOS resources for µC/OS-II as components. THINK, however, does not support the trace functionality to confirm the state of components or calling components without modification of C source code.

V. CONCLUSIONS

This paper has described the application of TECS to a two-wheeled inverted pendulum balancing robot. We have proposed the factory statement that is used to generate a kernel configuration file of an RTOS. There are three benefits of using RTOS resources as component: configurability the kernel of the RTOS, understandability the software structure, reusability components.

Moreover, the trace functionality has been introduced to confirm the state of cells or calling cells without modification of C source code. Moreover, the case study has shown understandable software structure.

REFERENCES


