UDC 004.41(075.8)

Siberian Journal of Science and Technology. 2017, Vol. 18, No. 4, P. 744–747

TO THE QUESTION ON IMPLEMENTATION OF MULTI-VERSION EXECUTION ENVIRONMENT SOFTWARE OF ONBOARD AUTONOMOUS UNMANNED OBJECTS BY MEANS OF REAL-TIME OPERATING SYSTEM

I. V. Kovalev, V. V. Losev, M. V. Saramud*, D. I. Kovalev, M. O. Petrosyan, V. V. Brezitskaya

Reshetnev Siberian State University of Science and Technology
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russian Federation
*E-mail: msaramud@gmail.com

The article deals with the functional and algorithmic implementation of multi-version execution environment of modules as components of the onboard software of autonomous pilotless objects by means of real-time operating system. One of the approaches to implementation multi-version execution environment – implementation of the principle of a pseudo-parallelism (imitation of concurrent execution of tasks by dividing the time of their execution) are given. Messaging process between multiple tasks implemented by cycloram, as the procedure of returning of voting result by queuing mechanism.

Keywords: multi-version programming, reliability, real-time operating system, execution environment.

Сибирский журнал науки и технологий. 2017. Т. 18, № 4. С. 744–747

ИСПОЛЬЗОВАНИЕ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ОПЕРАЦИОННОЙ СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ ДЛЯ РЕАЛИЗАЦИИ МУЛЬТИВЕРСИОННОЙ СРЕДЫ ИСПОЛНЕНИЯ БОРТОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВТОНОМНЫХ БЕСПИЛОТНЫХ ОБЪЕКТОВ

И. В. Ковалев, В. В. Лосев, М. В. Сарамуд*, Д. И. Ковалев, М. О. Петросян, В. В. Брезицкая

Сибирский государственный университет науки и технологий имени М. Ф. Решетнева
Российская Федерация, 660037, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
*E-mail: msaramud@gmail.com

Рассматривается функциональная и алгоритмическая реализация мультиверсионной среды исполнения как компонентов встроенного программного обеспечения автономных беспилотных объектов с помощью операционной системы реального времени. Дается один из подходов к реализации мультиверсионной среды исполнения – реализация принципа псевдопараллельности (имитация одновременного выполнения задач путем деления времени их выполнения). Процесс обмена сообщениями между несколькими задачами, представленный в виде циклограммы, использует в качестве средства обмена информацией механизм очередей.

Ключевые слова: мультиверсионная среда исполнения, голосование, задача, циклограмма, сообщения, очереди, надежность.

Introduction. The task of designing and software implementation of multiversion on-board software (OBS) of autonomous unmanned objects (AUO), is not trivial [1; 2]. At the same time, the task of forming a multiversion environment is no less important. There are a number of conditions necessary to ensure the operation of such an environment. Firstly, the implementation of an adequate mechanism of voting [3]. Secondly, to ensure data exchange between software modules. Thirdly, to provide a mechanism for extrusion of module in case that a similar decision is made by the arbitrator, as well as the subsequent addition of a new module, to maintain the specified reliability indicator of OBS AUO.

Search of solution. One of the approaches to implementation multi-version execution environment – implementation of the principle of a pseudo-parallelism. For example – the imitation of parallel execution of tasks by dividing their execution time. Such a functional principle is realized by the real-time operating system RTOS (real-time operating system), namely one of the versions – FreeRTOS [4] ported. This version is portable i. e. adapted for execution on the SoC (System on a Chip) [5].

Consider the schedule (fig. 1) of process of message exchange between multiple tasks implementation. Exchanging process implement procedure of voting result return by queuing mechanism implemented by FreeRTOS.

Task-receiver is a task that realizes the collection of data from *N*-version modules (Task-1, Task-2, …, Task-*N*) with the purpose of subsequent decision by the arbitrator, however, in the described process performing only the receiver function [5; 6].

Task-1, -2 – these are tasks that interpret *N*-version modules [7; 8], which implement functions of OBS AUO.

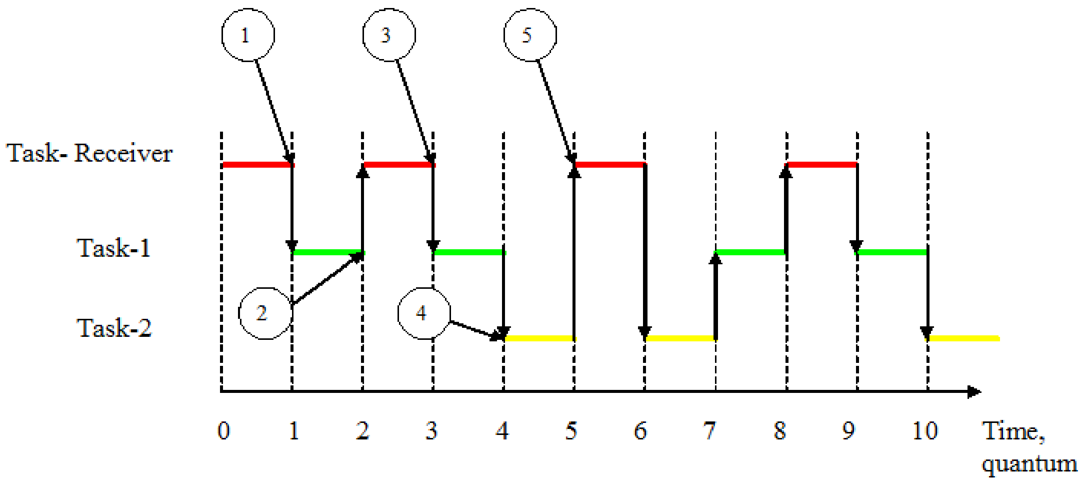

Fig. 1. Cyclogram of message exchange process between multiple tasks

Рис. 1. Циклограмма процесса обмена сообщениями между задачами

Algorithmizing the messaging process. The time component of the given cyclogram is distributed over equal time quanta.

Time point "0" – Initialized launch of FreeRTOS planner, which uses the "execution" status of the task with the highest priority. In this instance it is Task-Receiver (Com. API-function vTaskSrartScheduler() is responsible for the launch of the scheduler, API-function xTaskCreate() is responsible for task creation).

Time point "1" – Task-Receiver initiates an attempt to read an item from the queue, but goes into the "blocking" state, since the queue is empty at the time it was created. In this state Task-Receiver resides until the data occur in the queue, or up to the moment of time-out (120 ms). The next stage is the transition to the "execution" state of one of the Task-transmitters. (Task-1 or Task-2). It is not possible to declare, which task will go into this state, because each of them has equal priority. Let's suppose, this is Task-1 (Com. API-function vTaskSuspend()and vTaskResume() are respectively responsible for transition to the "blocking" state and exit from it).

Time point "2" – Task-1 writes a value "25" in the created empty queue. There is a return from "blocked" state at this point. Main function of this task is to capture data from the queue, and its priority is the highest (Com. API-function xQueueCreate(), is responsible for queue creation. To write an element to the end of the queue – FIFO principle implementation, API-function xQueue-SendToBack() is used, to write an element at the beginning of the queue – FIFO principle implementation, API-function xQueueSendToFront() is used).

Time point "3" – After accessing the queue and reading the data from it, the Trader-receiver is blocked again, since the queue is now empty. Control returns to the interrupted Task-1, which runs by call scheduler API-function taskYIELD() (Com. API-function xQueueReceive(), is responsible for reading element followed by removing it from the queue).

Time point "4" – The scheduler converts to the "execution" state an equitable Task-2, which, in turn, records the value of "50" in the queue.

Time point "5" – Leaves the "blocking" state of the high-priority Task-receiver and reads the data from the queue. Iteration of the loop happens next (Com. The current value of the time quantum counter may be received through API-function *xTaskGetTickCount()*).

Status of tasks during the system operation. The software system running under FreeRTOS, consists of a set of tasks. Tasks carried out separately, and in their own context. The scheduler controls the execution order, the change of tasks and their contexts. Each task has its own stack. Thread context is stored in stack extracted from the stack, when you pause and resume execution, respectively. Fig. 2 shows possible state of tasks. New ready-made task is created by command xTaskCreate(). The scheduler provide it with a time quantum for execution. The scheduler can leave thread in "ready" state if it is possible, taking into account the priorities or until such an opportunity arises. Also it is possible to send to the "blocking" state with the vTaskSuspend() command, from which only the scheduler can issue it by the vTask-Resume() command. From the "execution" state, the task can be transferred back to the "ready" state if it has not finished its work. However, the scheduler can decide to switch the task. For example, if a task with a higher priority has switched to a "ready" state. If the task needs to wait (the appearance of the element in the queue, the release of the mutex, the switching of the traffic light, and so on), then it is translated into the "waiting" state. Which can also be done by the command vTaskDelay(),setting the timeout after which the task goes into the "ready" state. When the expected event appears, the thread will be transferred to the ready state. By vTaskSuspend() command you can send the task from "execution" state to "blocking" state. If several tasks are in the "ready" state, the scheduler executes the task with the highest priority. Herewith tasks with lower priority are executed only after the higher priority task does not go into the "waiting" or "blocking" state. Task in the "waiting" or "blocking" state do not spend CPU time. The main difference between them is that the task in the "blocking" state does not respond to any events. And can be removed from this state only by the vTaskResume () command manually.

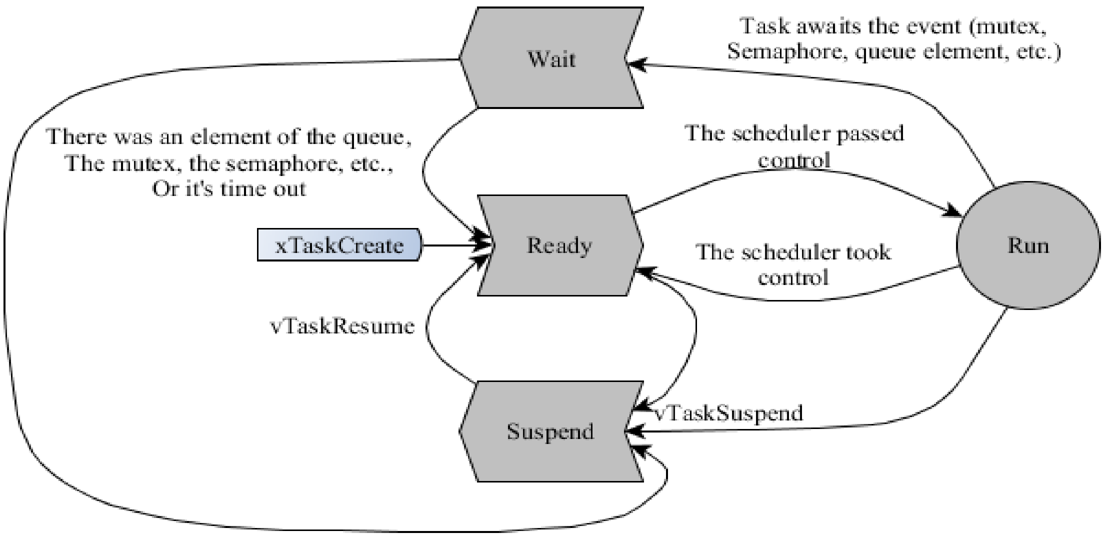
Wait
Task awaits the event (mutex, Semaphore, queue element, etc.)
There was an element of the queue, The mutex, the semaphore, etc., Or it's time out
The scheduler passed control
xTaskCreate
Ready
The scheduler took control
Run
vTaskResume
vTaskSuspend
Suspend

Fig. 2. Possible state of tasks

Рис. 2. Возможные состояния задач

Conclusion. This example demonstrates the messaging mechanism. According to which, value, placed in queue by task, represented by constant. This task interprets *N*-version modules (Task-1, -2). Returned by the module as a result of voting true value, demands additional algorithmization of *N*-version module [9; 10]. While the value generated by the module can take a different format, including the format of both boolean and integer variables [11], depending on voting mechanism [12]. Thus, the messaging mechanism, together with the scheduler, and the priority system implemented by FreeRTOS tools, allow you to build more flexible algorithms of voting, able to vary the weights of *N*-version modules, and as a consequence, affect the reliability of the multiversion software [13], including for the implementation of a multiversion environment for the execution of OBS AUO. In addition, thanks to the priority system, implementation of algorithm for changing the priority of each module, depending on its weight (reliability rating) is possible. This solution in conditions of limited resources will ensure the priority execution of the most error-free versions. Moreover, improve the reliability of the multiversion system in cases where, due to the limitation of the computing resources of the hardware platform, it is impossible to execute all versions within a specified period. Thus, the most reliable versions will not only have more weight in voting, but will be performed first, also. With the help of the priority system, the time limit for issuing answers to versions is limited. Task-receiver with the highest priority through the given time quantum reads all the responses from the queue and deletes (vTaskDelete ()) versions that did not have time to respond in time.

As a result of further analysis of mechanisms of the functioning of FreeRTOS and other real-time operating systems, one can note potential problems:

1. Inversion of priorities – a situation in which Traid-1, which has priority higher than Task-2, expects completion of its work, since Task-2 was the first to capture the mutex. The peculiar feature of the mutex capture is that only the thread that captures the mutex takes the decision to release it. The mutex release occurs only when the thread does its work, regardless of its duration and the priority value of other threads.

2. Interlocking – The situation is possible when several tasks need to access the same resources for its completion:

Task-1 is executed, captures mutex A, its performance is interrupted by Task-2, captures mutex B, tries to capture mutex A, because it is busy, switches to standby, Task-1 continues its execution, tries to capture mutex B, since it is busy, it's also goes into waiting. As a result, both tasks will not release needed mutexes and will be in the standby mode, effectively blocking each other.

We can exclude the occurrence of mutual interlocking during system design. Moreover, we can exclude the inversion of priorities or minimize the negative consequences when it occurs.

References

1. Kovalev I., Semenko T., Tsarev R. *Metodologiya otsenki i povysheniya nadezhnosti programmno-informatsionnykh tekhnologiy i struktur* [The assessment methodology and improve the reliability of software and information technologies and structures]. Krasnoyarsk, Feder. agentstvo po obrazovaniyu, Krasnoyar. gos. tekhn. un-t Publ., 2005, 160 p.

2. Kovalev I. V. Multiversion environment creation for control algorithm execution by autonomous unmanned objects. *IOP Conference Series: Materials Science and Engineering V International Workshop on Mathematical Models and their Applications.* 2016 7–9 November 2016, Krasnoyarsk, Russia. Vol. 173 (2017), 012025.

3. Kovalev I. [Analysis of problems in the field of research of software reliability: the multistage and the architectural aspect]. *Vestnik SibGAU.* 2012, No. 3 (55), P. 78–92 (In Russ.).

4. Stel'makh V. O., Kovalev I. V. [Building fault-tolerant control systems based on the multiversion approach]. *Materialy vserossiiskoi molodezhnoi konferentsii "Informatsionno-telekommunikatsionnye sistemy i tekhnologii (ITSIT-2012)"*. 2012. P. 172–173.

5. Kovalev P. V. [Graphoanalytical method for analyzing the multiversion software architectures]. *Mezhdunarodnyi zhurnal prikladnykh i fundamental'nykh issledovanii, Akademiya estestvoznaniya*. 2009. No. 6, P. 70 (In Russ.).

6. Kovalev I., Zelenkov P., Ognerubov S. The minimization of inter-module interface for the achievement of reliability of multi-version software IOP Conference Series: Materials Science and Engineering 17. XVII International Scientific Conference "Reshetnev Readings" 2015. P. 012006.

7. Kovalev I. V., Zelenkov P. V., Tsarev M. Y. The control of developing a structure of a catastrophe-resistant system of information processing and control IOP Conference Series: Materials Science and Engineering 17. XVII International Scientific Conference "Reshetnev Readings" 2015. P. 012008.

8. Kovalev I. V. [Multiversioned views method for increasing software reliability information and telecommunication technologies in corporate structures]. *Telekommunikacii i informatizaciya obrazovaniya*. 2003, No. 2, P. 50–55 (In Russ.).

9. Kovalev I. V., Dgioeva N. N., Slobodin M. Ju. The mathematical system model for the problem of multiversion software design Proceedings of Modelling and Simulation, MS'2004 AMSE : Intern. Conf. on Modelling and Simulation, MS'2004. AMSE, French Research Council, CNRS, Rhone-Alpes Region, Hospitals of Lyon. Lyon-Villeurbanne, 2004.

10. Kovalev I. V., Slobodin M. Ju., Stupina A. A. [Mathematical formulation of the problem of designing *n*-version software systems]. *Problemy mashinostroenija i avtomatizacii*. 2005, No. 3, P. 16–23 (In Russ.).

11. Engel E. A., Kovalev I. V. [Information processing using intelligent algorithms by solving wcci 2010 tasks]. *Vestnik SibGAU*. 2011, No. 3, P. 4–8 (In Russ.).

12. Kovalev I. Evaluation of the reliability of ACS with blocking protection modules. *The Devices.* 2013, No. 6, P. 20–23.

13. Saramud M. V., Zelenkov P. V., Kovalev I. V., Kovalev D. I., Bresizkaja V. V. Characteristics of software module reliability XVII International Scientific Conference "Reshetnev Readings" 2015, P. 87–88.

Библиографические ссылки

1. Ковалев И. В., Семенько Т. И., Царев Р. Ю. Методология оценки и повышения надежности программно-информационных технологий и структур / Федер. агентство по образованию ; Краснояр. гос. техн. ун-т. Красноярк, 2005. 160 с.

2. Multiversion environment creation for control algorithm execution by autonomous unmanned objects / I. V. Kovalev [et al.] // IOP Conference Series: Materials Science and Engineering V International Workshop on Mathematical Models and their Applications 2016 (7–9 November 2016, Krasnoyarsk). 2017. Vol. 173. P. 012025.

3. Ковалев И. В. Анализ проблем в области исследования надежности программного обеспечения: многоэтапность и архитектурный аспект // Вестник СибГАУ. 2012. Вып. № 3 (55). С. 78–92.

4. Стельмах В. О., Ковалев И. В. Построение отказоустойчивых систем управления на основе мультиверсионного подхода // Информационно-телекоммуникационные системы и технологии (ИТСИТ-2012) : материалы Всерос. молодеж. конф. 2012. С. 172–173.

5. Ковалёв П. В. Графоаналитический метод анализа мультиверсионных архитектур программного обеспечения // Международный журнал прикладных и фундаментальных исследований. 2009. № 6. С. 70.

6. Kovalev I., Zelenkov P., Ognerubov S. The minimization of inter-module interface for the achievement of reliability of multi-version software // Reshetnev Readings: IOP Conference. Series: Materials Science and Engineering, 17. 2015. P. 012006.

7. Kovalev I. V., Zelenkov P. V., Tsarev M. Y. The control of developing a structure of a catastrophe-resistant system of information processing and control // Reshetnev Readings: IOP Conference. Series: Materials Science and Engineering, 17. 2015. P. 012008.

8. Ковалев И. В., Юнусов Р. В. Мультиверсионный метод повышения программной надежности информационно-телекоммуникационных технологий в корпоративных структурах // Телекоммуникации и информатизация образования. 2003. № 2. С. 50–55.

9. Kovalev I. V., Dgioeva N. N., Slobodin M. Ju. The mathematical system model for the problem of multiversion software design // International Conference on Modelling and Simulation, MS'2004. AMSE / French Research Council, CNRS, Rhone-Alpes Region, Hospitals of Lyon. Lyon-Villeurbanne, 2004.

10. Ковалев И. В., Слободин М. Ю., Ступина А. А. Математическая постановка задачи проектирования *n*-версионных программных систем // Проблемы машиностроения и автоматизации. 2005. № 3. С. 16–23.

11. Engel E. A., Kovalev I. V. Information processing using intelligent algorithms by solving wcci 2010 tasks // Вестник СибГАУ. 2011. № 3. С. 4–8.

12. Оценка надежности АСУ с блокирующими модулями защиты / И. В. Ковалев [и др.] // Приборы. 2013. № 6. С. 20–23.

13. Характеристика надежности программных модулей / М. В. Сарамуд [и др.] // Материалы XIX Междунар. науч.-практ. конф., посвященной 55-летию Сибирского государственного аэрокосмического университета имени академика М. Ф. Решетнева / СибГАУ. Красноярск, 2015. Ч. 2. С. 87–88.