ASTES

Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 4, 385-394 (2021)

www.astesj.com

Special Issue on Innovation in Computing, Engineering Science & Technology

ASTESJ

ISSN: 2415-6698

A Scheduling Algorithm with RTiK+ for MIL-STD-1553B Based on Windows for Real-Time Operation System

Jong-Jin Kim, Sang-Gil Lee, Cheol-Hoon Lee*

Department of Computer Engineering, Chungnam National University, Daejeon, 305764, Korea

ARTICLE INFO

Article history:
Received: 01 July, 2021
Accepted: 09 August, 2021
Online: 26 August, 2021

Keywords:
RTX
RTiK+
APIC
Tablet PC
Inspection Equipment
Real-Time Operating System
Windows
MIL-STD-1553B
RM Scheduling Algorithm

ABSTRACT

In devices using Windows operating system based on x86 system, the real-time performance is not guaranteed by Windows. It is because Windows is not a real-time operating system. Users who develop applications in such a Windows environment generally use commercial solutions such as the RTX or the INtime to provide real-time performance to the system. However, when using functions and API for simple real-time processing, an issue of high development cost occurs in terms of cost-effectiveness.

In this paper, the RTiK+ was implemented in the type of a device driver by controlling the MSR_FSB_FREQ register to generate a timer interrupt independent of Windows in the Windows 8 and providing a real-time functionality to the user mode by re-setting the local APIC count register. And the operating frequency of the CPU is changed to minimize power consumption for battery life in a mobile device such as a Tablet PC.

In particular, the weapon system uses highly reliable MIL-STD-1553B communication and performs BC and RT functions of MIL-STD-1553B to transmit and/or receive data in communication between component and component. It is significantly importance to guarantee integrity of data without loss data during communication. For this purpose, it is proposed to implant the Scheduling algorithm with the RTiK+ for MIL-STD-1553B communication for Windows 8 to support a real-time processing in the Windows operating system on the embedded system, and to use the periods of 2ms (max), 5ms and 10ms provided by the RTiK+ for real-time processing when performing the BC and RT functions of MIL-STD-1553B communication. In this paper, the method of the scheduling algorithm with RTiK+ for MIL-STD-1553B to provide real-time processing is proposed for the Windows based on x86 system.

1. Introduction

1.1. Research Background

Recently, with the development of hardware and embedded systems, various mobile devices such as Tablet PC and smart phone used in daily life are being used instead of dedicated equipment for special purposes according to the user environment. Therefore, it is necessary to support an accurate real-time processing function in order to transmit/receive accurate commands or data to a mobile device without data loss of information from the target equipment. And this paper is an extension of work originally presented in Real-time Processing Method for Windows OS Using MSR_FSB_FREQ Control [1-5].

In general, a real-time system should be predictable in any situation, which means that time deterministic should be guaranteed. In this paper, to ensure time determinism, the MSR_FSB_FREQ register, which determines the operating frequency of the CPU to provide real-time processing performance in the Windows 8 environment, is controlled and the local APIC timer count register value is reset to operate a window independent timer. A study was conducted to guarantee the periodicity set by the user. Also, when checking the function and performance of a guided weapon system using a window-based inspection equipment in a guided weapon that uses fast MIL-STD-1553B communication such as 2ms between components, The RTiK+ is first installed to provide real-time performance to the Windows 8, and then it use the period provided by RTiK+ for scheduling algorithm to meet deadline in MIL-STD-1553B communication in guided weapons system, and also a study for miss rates to

*Corresponding Author: Prof. Cheol-Hoon Lee, Department of Computer Engineering, Chungnam National University, Korea, Email: clee@cnu.ac.kr

guarantee data integrity without loss is conducted through experimental tests to be proofed.

1.2. Research Purpose

In the case of a tablet PC, The Windows as operating software is mainly used as an operating system to provide compatibility with libraries that have been developed and operated, dependency on the device's operating system, and extensibility for various applications. To support a real-time processing function in the tablet PC, a commercial solution that is installed in addition to the Windows operating system installed in the tablet PC and provides a real-time functionality must be used. Such products include IntervalZero's RTX, which is currently widely used. The purchase price is very high, and when installed in equipment for mass production, royalty and maintenance costs are high, which causes a high development cost burden on developers. To improve this matter, The RTiK+ was designed to provide a real-time performance in Windows 8 based on x86 system.

In addition, the Windows operating system used for inspection equipment prioritizes compatibility, stability and reliability due to the characteristics of the weapon system, so older versions such as Windows 7 and Windows 8 are used rather than the latest versions such as Windows 10 [6]. For example, the flight body inspection equipment that recently developed an unmanned aerial vehicle developed in 2017 uses the Windows 7 released in 2009 [7].

In this paper, the RTiK+ is designed as a method by controlling the MSR_FSB_FREQ register to generate a window-independent timer interrupt in Windows 8 and providing a real-time period to the user mode by resetting the counter register of the local APIC, and the CPU operating frequency. A study how to provide a real-time performance by accessing the MSR_PKG_CST_CONFIG_ CONTROL register that determines the C-States so that the frequency controlling the C-States does not change.

A MIL-STD-1553B communication with high reliability is used mainly in the precision guided weapon system [8], and the proposed RTiK+ to provide a real-time processing function in the Windows operating system of the equipment is implanted in Windows 8, and it supports the BC and RT function of the MIL-STD-1553B to provide a real-time processing. The scheduling algorithm with RTiK+ supports a real-time period to the BC and RT function when performing communication.

Finally, an evaluation for experiment results is conducted to proof the miss rate that complies with the deadline in the set period in the MIL-STD-1553B communication which RTiK+ for a real-time processing is applied. The RTiK+ to provide a real-time processing is implanted in the Windows 8 based on x86 system, and the scheduling algorithm is applied to perform the MIL-STD-1553B communication. The integrity is verified using the RDTSC for measurement of the period of a real-time function and PASS 3200 for data monitoring of the MIL-STD-1553B, and the miss rate of the scheduling algorithm for the MIL-STD-1553B is analyzed. And to see if the RTiK+ can be used as a replacement item for the RTX, a real-time performance is measured and analyzed using the same experimental configuration for measuring performance of the RTX, showing that the RTiK+ can be used as a substitute for a third party.

The paper consists of chapters 1 to 6. Chapter 2 studies RTX, window scheduling, RM algorithm, and MIL-STD-1553B. Chapter 3 describes the real-time processing in the tablet PC proposed in this paper, and the real-time performance implementation using the MIL-STD-1553B communication scheduling algorithm. Chapter 4 describes the experimental environment, composition, and results. Chapter 5 analyzes the algorithm and results, Chapter 6 concludes by describing the summary of this paper and future research.

2. Related Research

2.1. RTX

IntervalZero released the RTX operating in kernel mode (ring 0) based on x86 computers in 1995. The RTX is software that provides a real-time performance to support a real-time function in Windows XP or Windows 7 but there is no solution for Windows 8 currently. The RTX provides a development environment that is easy to access and familiar to developers who use Windows based on a real-time operating systems, which are the advantages of Windows [9].

2.2. Local APIC

An APIC is an interrupt controller provided by the x86 architecture. It implements the function of delivering a hardware interrupt to the interrupt descriptor table with the address of the interrupt handler. A local APIC is an interrupt controller provided by the x86 architecture as an extension of the programmable interrupt controller [10-13].

2.3. Scheduling of Process and Thread in Windows

The Windows operating systems based on x86 system have processes and threads to process specific tasks. In general, the threads are divided into user-level threads and kernel-level threads. A kernel-level thread is used in Windows, and in Windows, a thread is assigned in a processor by a kernel scheduler and implements a scheduling. It is a scheduling algorithm that is scheduled according to the priority of all threads created in the operating system regardless of the priority of the process. The Windows scheduler uses a round-robin scheduling method. A process has six priorities, and a thread has seven priorities, and consists of a total of thirty-one priorities from 0 to 31 [14].

2.4. RM Scheduling Algorithm

The RM Scheduling, The RM means Rate-Monotonic, for a real-time system is a scheduling algorithm that gives the highest priority to the shortest period. The operating system to which this algorithm is applied is generally called preemptive and has a decisive characteristic for response time [15]. When there are n processes, the upper limit of total utilization on the system can calculate using the equation (1) [16]. If the total system utilization is less than or equal to the utilization U_{RM} calculated by the equation (1), it proves that scheduling is possible with the RM scheduling algorithm.

$$U_{RM}(n) = n\left(2^{\frac{1}{n}} - 1\right) \quad (1)$$

The RM scheduling algorithm is a method of fixed priority scheduling in which a fixed priority is given to each task, and when a task having a higher priority arrives, the currently executing task is preempted. The RM algorithm determines the priority based on the period of the task, and it is an algorithm that gives a high priority to a task with a short execution period [17-19].

2.5. C-States

It can command the CPU to enter a low-power mode to save battery when the CPU is idle. Each CPU has multiple power modes, collectively known as the C-States. In general, in the case of a mobile device such as tablet PC in the x86 architecture, the low-power technique provides five states of the CPU called the C-States as shown in Table 1 [20].

Table 1: C-States mode in Intel's CPU

Mode	Status of CPU
C0	Processor state: Full On
C1	Processor state: Auto-Halt
C1E	Processor state: Auto-Halt
C3	Processor state: Deep Sleep
C6	Processor state: Deep Power Down

2.6. MIL-STD-1553B

The MIL-STD-1553B, which is mainly applied to aircraft weapon systems and guided weapon systems, has a maximum transmission speed of 1Mbps, and it is a half-duplex and a serial communication method.

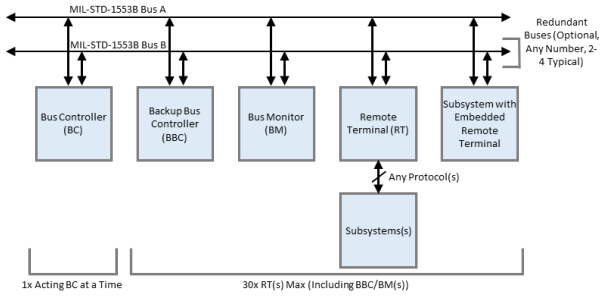


Figure 1: Interface Diagram of MIL-STD-1553B bus

Also, there is a restriction that the length of data that can transmit or receive data at a time is limited to a maximum of 32 words by a protocol. As shown in Figure 1, a bus controller (BC), a remote terminal (RT) and a monitoring (MT) are interfaced to the MIL-STD-1553B bus. The main function of the BC is to control the transmission and reception of all RTs on the bus. The RT has a function of transmitting or receiving data by responding to commands from the BC, and the MT has a function of monitoring all data moving on the bus [21, 22].

3. Methodology

3.1. Overview

In Windows 8, the initial count register (FFF0 0380h), as a local APIC timer register, is initialized to 0 after the operating system is booted. Therefore, since it is impossible to utilize a timer interrupt independent of the Window, there is a matter in providing a real-time processing function corresponding to the computer's system clock. In addition, The C-States are used to increase battery operation time by reducing power consumption of mobile device. The C-States function has the problem that the system clock is changed frequently flexibly, and an accurate period calculation for guarantying a real-time is not allowed [23, 24].

In this paper, in order to solve this issue, The RTiK+ is designed as shown in Figure 2, which uses a timer interrupt independent of the Windows to provide a real-time processing functions in the Windows 8 to support real-time performance for threads in the user mode. And the scheduling algorithm to support a real-time performance for MIL-STD-1553B communication is implemented based on RTiK+.

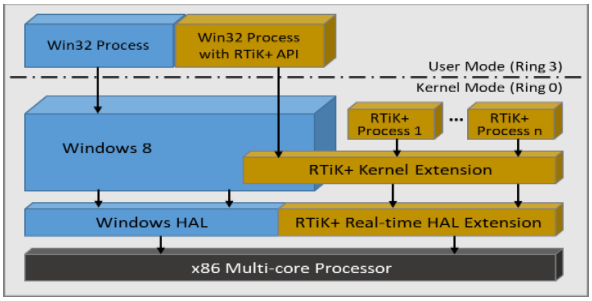


Figure 2: Architecture of RTiK+ for Real-Time Operating System

3.2. Local APIC Timer Control

As shown in Figure 2, The RTiK+ supplies a timer interrupt independent of the window through Kernel Resources access. And as shown in Figure 3, The RTiK+ is implemented to provide a real-time functionality in the Windows using the local APIC and kernel resources of the application processor [25, 26].

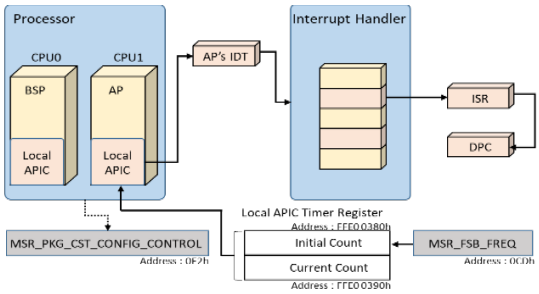


Figure 3: A Real-Time Processing Operating Process of the RTiK+

Before the timer interrupt is occurred, the initial counter register (FEE0 0380h) of the local APIC of the application processor sets the initial counter register value according to the value set in the MSR_FSB_FREQ register (0CDh) to generate a timer interrupt independent of the Windows. For such interrupt processing, a real-time task is created by registering an interrupt object in the interrupt descriptor table (IDT). In addition, The RTiK+ guarantees the periodicity of the real-time thread by using the MSR_PKG_CST_CONFIG_CONTROL register and minimizes the period error tolerance of the set task by controlling the C-States operation mode of the CPU to ensure the periodicity of the interrupt for thread processing [27-29].

3.3. Time Interrupt Control through FSB

As shown in Figure 4, x86-based system has two chipsets called the Northbridge and the Southbridge, the Northbridge is

between memory controller hub and the memory/graphic card slots, and the Southbridge is between I/O controller hub and PCI slots.

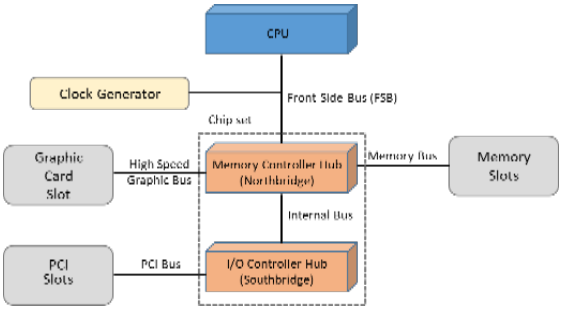


Figure 4: A Block Diagram of Chipset based on x86 Architecture [30]

A FSB also provides the ability to use a clock multiplier to determine the operating speed of the CPU and it provides synchronization by affecting the memory clock speed. The chipset block diagram of the x86 architecture is as shown in Figure 4, and the frequency generated by the clock generator is connected to the FSB and provided by the CPU and set as the initial counter register (FFF0 0360h) value of the timer register of the local APIC and then involves in determining the generation period of the timer interrupt. And in the case of the Windows 8, the operating frequency of the FSB according to the CPU workloads is provided through the lower three bits of D2:D0 of the MSR_FSB_FREQ register (0CDh), which is a hardware resource.

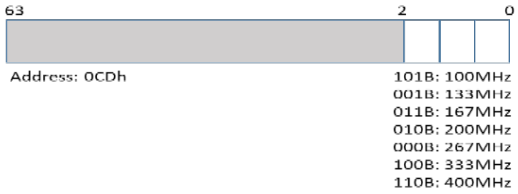


Figure 5: A MSR_FSB_FREQ Register

Therefore, as shown in Figure 5, to set the timer interrupt generation period of the local APIC to a fixed value, check the operating frequency of the currently executing FSB in the MSR_FSB_FREQ register, and set the local APIC timer to the value of the initial counter to generate an independent timer interrupt [31].

3.4. Time Deterministic through controlling C-States

Unlike the existing Windows 7, The Windows 8 provides a power saving function that reduces power consumption in real time by turning off power to some devices by adjusting the CPU clock low through access to the C-States for low-power control of the CPU. The following equation (2) is applied as the equation used to determine the CPU clock in Windows 8 [24, 32].

$$\text{CPU clock} = \text{FSB frequency} \times \text{Clock multiplier} \quad (2)$$

The FSB can control the power consumption by controlling the value of the clock multiplier by the Windows according to the hardware specifications. However, the method of adjusting the value of the clock multiplier to lower the clock of the CPU has a matter in that it is difficult to guarantee the time determinism of the timer interrupt of the local APIC. In addition, a problem occurs in that the aforementioned time determinism cannot be guaranteed because a delay time occurs due to the time when the CPU is activated from the idle state or sleep state.

In this paper, The C-States are controlled through the MSR_PKG_CST_CONFIG_CONTROL register (0E2h) provided by x86 system to guarantee the periodicity of the RTiK+. And each speed of the core clock according to the setting of the C-States is seemed as shown in Table 2. When the C-States are enabled, the clock is automatically adjusted based on the utilization rate the CPU is processing tasks. Also, when the C-States are disabled, the clock is kept at its maximum value regardless of the CPU utilization [33].

Table 2: A CPU Clock Status depending on the C-States

Type of CPU		C-States Enable	C-States Disable
Core 1	Standard	3199.86 MHz	3199.86 MHz
	Current	3199.86 MHz	3199.86 MHz
Core 2	Standard	1599.93 MHz	3199.86 MHz
	Current	1599.93 MHz	3199.86 MHz

3.5. Process in User Mode of the RTiK+

A study on the real-time processing function of the user mode provided in the existing Windows such as the Windows XP and the Windows 7 before the Windows 8 has a matter that the periodicity, data accuracy, and data integrity are not guaranteed because event signals are lost in an environment of multi-core processor. To solve this problem, the RTiK+ provides a real-time processing function in the user mode as shown in Figure 6.

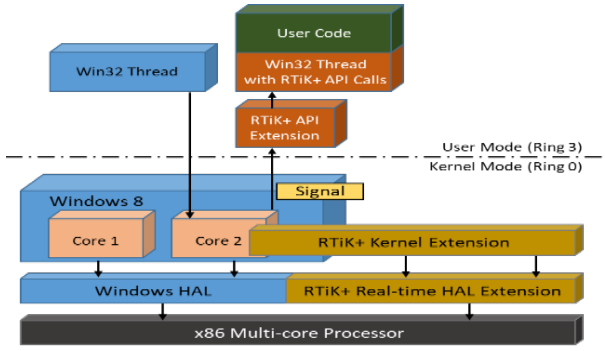


Figure 6: An Operation Process of the RTiK+ for a Real-Time Processing

The algorithm coded in the RTiK+ to set the priority of the process to the highest to guarantee data accuracy and data integrity is shown in Figure 7 [34]. It is implemented to call the Process_Affinity() function inside the real-time thread, and the real-time thread inside Process_Affinity(), by calling the SetProcessAffinityMask() function, an API provided by the Windows, the CPU is specially designated so that the real-time thread operates only on the core where RTiK+ operates.

Algorithm 1:

Result:

```
void Process_Affinity()
{
   SetPriorityClass(GetCurrentProcess(), REALTIME_PRIORITY_CLASS);
   SetProcessAffinityMask(GetCurrentProcess(), 0x02);
}
```

Figure 7: An Algorithm to set for Priority of Process

In addition, by calling the SetPriorityClass() function to set the priority of the real-time thread and process of RTiK+, the current real-time thread is designated as the highest level as a REALTIME_PRIORITY_CLASS, so that the CPU is not preempted by other Window's threads [35].

When the RTiK+ process is created, the process must be created by setting it to the REALTIME_PRIORITY_CLASS priority level, which is the highest priority. Figure 8 is the algorithm coded to provide a real-time processing function in the user mode of the RTiK+.

Algorithm 2:

Result:

```
void main()
{
    hThread = (HANDLE)_beginThreadex(NULL,0,ThreadFunction,NULL,0,(unsigned*)&dwThreadID);
    SetThreadPriority(hThread,THREAD_PRIORITY_TIME_CRITICAL);
}
```

Figure 8: An Algorithm to set for Priority of Thread

3.6. Priority-based Scheduling Algorithm

In the priority-based scheduling algorithm, a scheduling is implemented online, and when an event such as the creation or termination of a task occurs, the scheduling is performed first in the high-priority task. For this reason, the scheduling is performed while tasks are being performed, and there is an advantage of being able to respond more flexibly according to the state of the system, but on the other hand, it sometimes causes overhead in the operating system [36].

In this paper, the RM scheduling algorithm, a fixed-priority scheduling algorithm proposed by Liu and Layland, is applied to the RTiK+. The RM is an algorithm of giving a high priority to a task with a short period when all tasks are periodic tasks, the deadline coincides with the period, and the tasks are independent from each other. In this paper, the total utilization rate of the RM scheduling algorithm is U_{RM} (n) = n (2^(1/n)-1). If tasks are maximum 100, U_{RM} (100) = 0.69555. And no matter how many tasks are, the total utilization U_{RM} is within 0.69314.

3.7. A Process in User Mode of RTiK+

3.7.1. The BC Algorithm for a Real-Time Performance

In the MIL-STD-1553B, the BC function controls the direction of data when transmitting data through the bus, and transmits data to the RT by sending transmitter (Tx) and receiver (Rx) commands from the BC of the MIL-STD-1553B communication. In order to guarantee a real-time processing function of MIL-STD-1553B communication, it is implemented to operate the Tx and Rx commands in the real-time thread provided by the RTiK+. Figure 9 is the BC algorithm of MIL-STD-1553B communication that implements the Tx and Rx commands in the periodic thread of the RTiK+.

In this algorithm, Tx command releases the Rx link with the m1553_delete_link() function, connects the Tx link with the m1553_add_link_to_chain() function, sends the Tx command from the BC with the m1553_delete_link() function, and checks the data transmitted from the RT with m1553_read_link_data() function. And the Rx command releases the Tx Link with the m1553_delete_link() function, connects the RX Link with the m1553_load_chain() function, and transmits data to the RT with the m1553_load_chain() function.

Algorithm 3:

Result:

```
m1553_delete_link(device_number, RxLink.link_id, RxLink.chain_id);
status = m1553_add_link_to_chain( device_number, &TxLink );
if ( status == FAILURE )
{
    printf( "FAILURE\n" );
    printf( "%s\n", sbs_read_error() );
}

status = m1553_load_chain( device_number, 1 );
if ( status == FAILURE )
{
    printf( "FAILURE\n" );
    printf( "%s\n", sbs_read_error() );
}

uSleep(600);
m11553_read_link_data(device_number, TxLink.link_id, TxLink.buff_id[0], r_buffer);

m1553_delete_link(device_number, TxLink.link_id, TxLink.chain_id);
status = m1553_add_link_to_chain( device_number, &RxLink );
if ( status == FAILURE )
{
    printf( "FAILURE\n" );
    printf( "%s\n", sbs_read_error() );
}

m11553_write_link_data(device_number, RxLink.link_id, RxLink.buff_id[0], r_buffer);
status = m1553_load_chain( device_number, 1 );
if ( status == FAILURE )
{
    printf( "FAILURE\n" );
    printf( "%s\n", sbs_read_error() );
}
uSleep(600);
```

Figure 9: BC Algorithm for MIL-STD-1553B based on Real-Time Processing

3.7.2. The RT Algorithm for a Real-Time Performance

In the MIL-STD-1553B, the RT function transmits or receives data through the bus according to the BC commands. In order to check a real-time performance of the RTiK+, it is designed to store data in the buffer at the same period as the period set in the BC of the inspection equipment by the RTiK+ to send 32 words of the value increased by one (1) from the previous data. However, the matter of data loss due to the gap time for enabling the RTiK+ implanted in the BC and the RT function also occurs in the RT same as the BC. In order to solve this matter, a double buffer was used in the same way as the BC. Figure 10 shows the RT algorithm of the MIL-STD-1553B communication using double buffer.

Algorithm 4:

Result:

```
before_actptr_TX = actprt_TX;
actprt_TX = LL_get_ram( device_number, btrptr_TX + rt1sa2t.sa + 32);

If (before_actprt_TX != actptr_TX)
{
    if ( rt1sa2t.buff_id[0] == actprt_TX )
    {
        for (i=0; i<32, i++)
        {
            w_buffer[i] = (SBS16)( cnt );
        }

        m1553_write_sa_buffer( device_number, 32, rt1sa2t.buff_id[1], w_buffer);
        cnt++;
    }
    else if (rt1sa2t.buff_id[1] == actprt_TX
    {
        for (i=0, i<32, i++)
        {
            w_buffer[i] = (SBS16)( cnt );
        }

        m1553_write_sa_buffer( device_nmber, 32, rt1sa2t.buff_id[0], w_buffer);
        cnt++;
}
```

Figure 10: RT Algorithm for MIL-STD-1553B based on Real-Time Processing

3.7.3. Scheduling Algorithm of MIL-STD-1553B for a Real-Time Performance

When performing the MIL-STD-1553B communication, it takes time to activate the RTiK+ each time to utilize the RTiK+. A period error occurs in communication. Also, when a data transmission occurs in the BC due to the application of polling method applied in the paper and a data is received at the RT. In a state in which all data transmitted from the BC are not correctly received, the next data to be transmitted updates the buffer, resulting in a matter that correct data cannot be received at the RT. Therefore, in order to improve this problem, it is necessary to set the period of the RTiK+ for MIL-STD-1553B communication by considering the time it takes to transmit a data with the MIL-STD-1553B's BC command and the time it takes to receive data with the RT command. And a scheduling with RTiK+ for MIL-STD-1553B is required. Also, in the MIL-STD-1553 communication proposed in this paper, there is a problem in that it does not guarantee the periodic operation of data transmission from the BC to the RT because the Rx command is immediately sent after the Tx command is finished. In addition, since the RTiK+ operates at the period initially set, the period of the Tx command and the Rx command cannot be set respectively, so there can be a difference in the operating period of the RT and data loss can occur in this case. To solve this matter, when the Rx function is finished, the Rx Link is released, and the Tx Link is connected immediately to execute Tx function. The scheduling algorithm using the RTiK+ to guarantee a real-time processing performance when transmitting and receiving data in the MIL-STD-1553B communication was proposed.

This scheduling algorithm applied when performing the MIL-STD-1553B communication between the host PC and the target PC is assumed as follows.

- The RTiK+ uses the RM algorithm and preempts the CPU.
- The MIL-STD-1553B communication occurs periodically.
- Task period (Period, pi) and deadline (Deadline, Di) are the same. (pi = Di)
- The task start time (Release Time, Øi) is 0.
- The RTiK+ is implanted to both the host PC and target PC, and is activated only when the RTiK+ is utilized.
- The BC and the RT function are for MIL-STD-1553B, and interrupt cannot be used in an asynchronous method.
- The Tx and Rx command occur alternately. The data received by the Tx command is read from the buffer and transferred to the Rx command as it is.
- Transmits up to 32 words of data as in the Tx and Rx commands, and the transmission time is up to 0.80ms for all tasks [37].
- The Tx and Rx command cannot occur within a half of a period, therefore disconnect and create each time Tx to Rx transitions. Thus Tx(ei) > Rx(ei) or Tx(ei) < Rx(ei).
- The BC function and the RT function each use a double buffer.

Figure 11 shows the scheduling algorithm for the MIL-STD-1553B communication. The MIL-STD-1553B occurs when the Tx and Rx commands are alternated between the host PC and the target PC, and each task Ti (T0, T1, T2, T3, ..., Tn) has a maximum of 32 words of data. When transmitted, the execution time ei of each task is 0.80ms. At this time, since the deadline is the same as the period, it is scheduled so that transmission or reception of data within the period is normally completed without overhead. Also, when the MIL-STD-1553B communication is performed, RTiK+(pi), Tx(pi), Rx(pi), Buffer1(pi) and Buffer2(pi) are operated with an independent period each to match the period provided by the RTiK+.

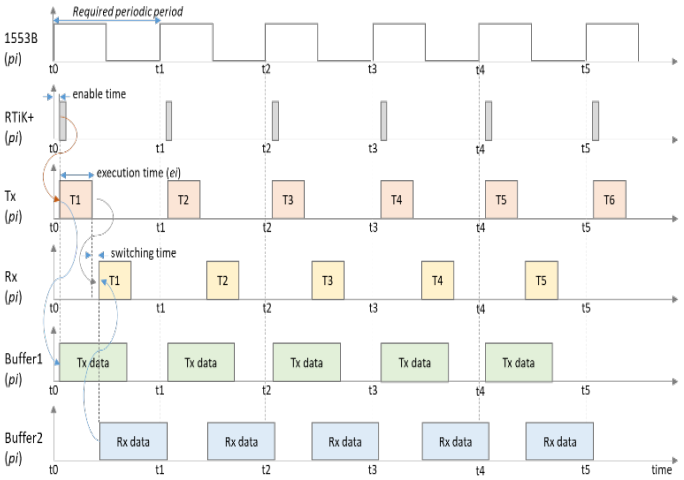


Figure 11: The Scheduling Algorithm with RTiK+ for MIL-STD-1553B

And if the task for the MIL-STD-1553B communication scheduling algorithm is as shown in Table 3, there are three Ti = (Øi, pi, ei, Di) tasks in the Tx and Rx commands linked to the RTiK+ pi period, verified if scheduling is possible through simulation that when the same 32 words of data transmission is occurred in the MIL-STD-1553B communication. That is, it was assumes that when this simulation is satisfied, all other communication conditions are also satisfied. At this time, when the Tx and Rx commands are executed, each task needs a maximum execution time of 0.80ms to transmit 32 words.

Table 3: A related Tasks of Tx/Rx for the MIL-STD-1553B Scheduling

Tasks	Release time (Øi)	Period (pi)	Execution time (ei)	Deadline (Di)
RTiK+ pi	0ms	2ms	0.01ms	2ms
Tx pi	0ms	2ms	0.80ms	2ms
Rx pi	0ms	2ms	0.80ms	2ms
T1	0ms	2ms	0.80ms	2ms
T2	0ms	2ms	0.80ms	2ms
T3	0ms	2ms	0.80ms	2ms

Using this method of the scheduling algorithm, each T1, T2, T3, …, Tn while Tx and Rx commands are being executed. It shows that the BC function of the MIL-STD-1553B can be implemented without data loss within the period set in the RTiK+ by the tasks. If the equation (3) is applied to calculate the system utilization rate for Tx(pi) and Rx(pi), the system utilization rate is 0.80/2 + 0.08/2 = 0.80, and the total utilization rate U_{RM} is 0.82 by applying the equation (1). It can be seen theoretically that the MIL-STD-1553B scheduling algorithm using the RM algorithm

operates normally with the system utilization rate (0.80) < U_{RM} (0.82).

Therefore, in the scheduling algorithm proposed in this paper, even if a Tx command executes in the MIL-STD-1553B communication and an Rx command executes immediately, the MIL-STD-1553B communication in real time meets the deadline without data loss within the period set in the RTiK+.

4. Results

4.1. Experimental Environment

To verify the real-time performance of the scheduling algorithm with the RTiK+ for MIL-STD-1553B communication, the experimental specifications for the host PC and the target PC is shown in Table 4.

Table 4: An Experimental Environment between host PC and target PC

Item	Host PC	Target PC
CPU	Intel Pentium Dual-Core 2117U @1.80 GHz	Intel Core i5-2500 @3.30 GHz
OS	Windows 8	Windows 7

4.2. Experimental Method

A verification is a measurement method that calculates the period by storing data using the RDTSC command to get a timestamp value to proof a real-time processing performance of the RTiK+. The algorithm coded in RDTSC for this experiment is shown in Figure 12.

The current timestamp stored in 64-bit values of EAX and EDX is read, store the timestamp value in the timearray[i].QuadPart array at i index, and calculate the difference between adjacent index values in the array. The f file is designed to be saved. In this way, a timer interrupt set with a period of 2ms occurs in the RTiK+, and whenever a real-time thread is executed in the user mode, the value of the timestamp is stored in the array.

Algorithm 5:

Result:

```
Hthread()
{
   _asm
   {
      RDTSC
      MOV lowval, EAX
      MOV highval, EDX
   }
   clockcal2_LowPart = lowval;
   clockcal2_HighPart = highval;
   timearray[i].QuadPart = clockval2.QuadPart;
   if(i==32000)
   {
      for(j=1, j<32000, j++)
      {
         fprint(f, "%.10f\n", (double)(timearry[i].QuadPart-timearray[j-1].QuadPart))/CPU);
      }
      i=0;
      break;
   }
}
```

Figure 12: An Algorithm with RDTSC for Period Measurement of the RTiK+

4.3. Measurement and Results

4.3.1. A Measurement for MIL-STD-1553B Communication

To verify a real-time performance support of the RTiK+ in the MIL-STD-1553B communication, the experimental environment was configured as shown in Figure 13. The RTiK+ is implanted to the host PC equipped with the MIL-STD-1553B communication, and the Period of the MIL-STD-1553B communication is set to 2ms with the RTiK+. The experimental data were measured with the RDTSC command as previously explained.

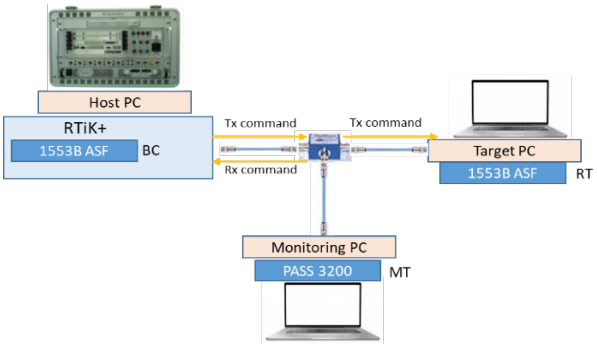


Figure 13: A Configuration to measure Period of MIL-STD-1553B

And to check the integrity of the data that there is no loss of transmitted and received data when performing the MIL-STD-1553B communication, The SBS Technologies' PASS 3200, a special-purpose MIL-STD-1553B monitoring equipment, is as shown in Figure 13, it connects between the host PC operating with the MIL-STD-1553B BC function and the target PC operating with the MIL-STD-1553B RT function to operate the MIL-STD-1553B MT function.

4.3.2. Result of Measuring Period during MIL-STD-1553B

In this paper, the maximum experiment period of 2ms was measured by RDTSC command during the MIL-STD-1553B communication. In addition, in order to compare a real-time performance for the MIL-STD-1553B between RTiK+ and RTX, the Period of the MIL-STD-1553B communication was measured as shown in Table 5 by applying the same experimental environment and experimental method.

Table 5: Comparison with Period of MIL-STD-1553B between RTiK+ and RTX

Period	Period of RTiK+			Period of RTX		
	Min.	Max.	Tolerance	Min.	Max.	Tolerance
2ms	1.993	2.031	1.57%	1.901	2.015	4.91%
5ms	4.955	5.003	0.90%	4.998	5.252	5.04%
10ms	9.993	10.005	0.06%	9.996	10.003	0.04%

4.3.3. A Result of Monitoring Data during MIL-STD-1553B

During the MIL-STD-1553B communication, in order to check the accuracy and integrity of the transmitted data, all data passing through the MIL-STD-1553B communication bus is stored with the PASS 3200 using the MIL-STD-1553B MT function as Figure 13. As shown in Figure 14, the first data transmitted by performing Tx command from the MIL-STD-1553B communication BC function to RT function is 32 words, all of which are 0xC912, and 32 words received by Rx command from RT function are all 0xC912. It seems that there is no loss or wrong data.

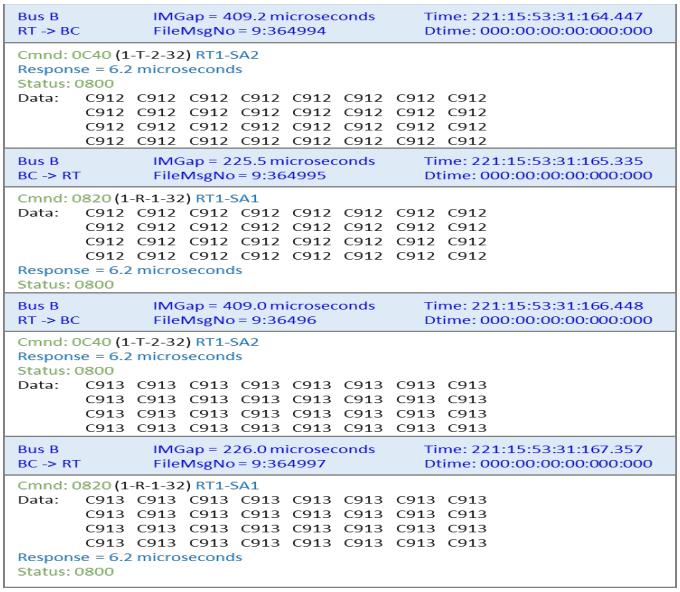


Figure 14: A Result of the MIL-STD-1553B monitoring by PASS 3200

5. Discussion

5.1. Process in User Mode of RTiK+

If the scheduling algorithm for the MIL-STD-1553B is analyzed based on the experimental results for period of the MIL-STD-1553B communication, Figure 15 and 16 is shown. As can be seen from this result, there was no case of overload the deadline in the 2ms period of the MIL-STD-1553B communication to which the scheduling algorithm with RTiK+ was applied, and it was proofed that the MIL-STD-1553B communication operates normally using the 2ms period of the RTiK+. As shown in Table 5, when the tolerance of the RTiK+ used in the MIL-STD-1553B communication is 1.57%, it is ±0.03ms from 2ms, but when the period for each task of Tx(pi) and Rx(pi) transmits 32 words, the maximum transmission time is 0.80ms, which is less than the half period of 2ms. The system utilization rate is 0.32 from the equation (3), and U_{RM} is calculated as 0.82 from the equation (1). It means that the scheduling algorithm for the MIL-STD-1553B operates normally without any communication errors.

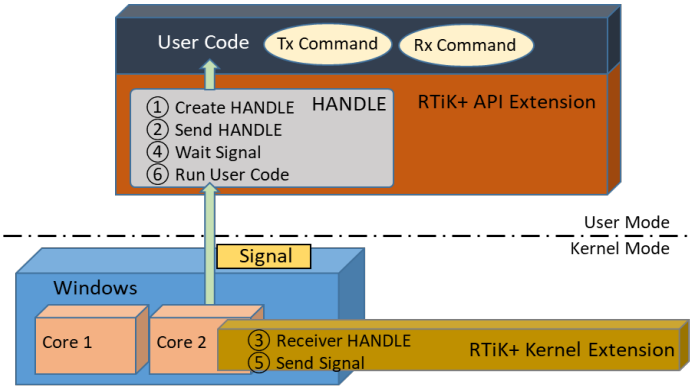


Figure 15: An Operation Diagram of Tx and Rx command for MIL-STD-1553B Scheduling Algorithm

In addition, this is the result of proving that the miss rate is 0 in the 2ms period provided by the RTiK+ for real-time processing function in the MIL-STD-1553B communication.

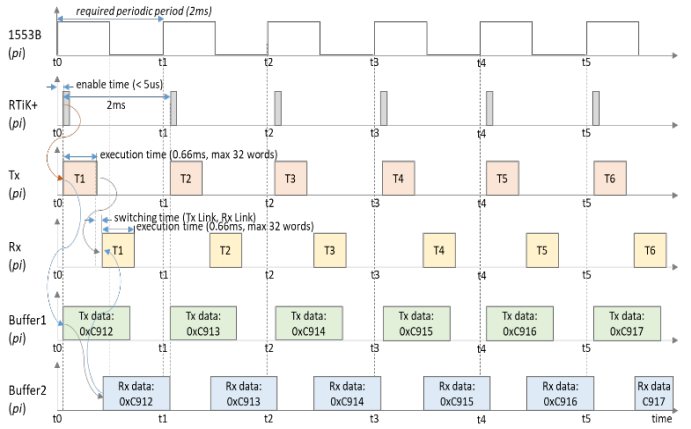


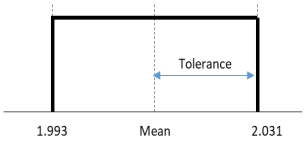
Figure 16: An Analysis Diagram for MIL-STD-1553B Scheduling Algorithm

5.2. Evaluation for Real-Time Performance of RTiK+

In order to evaluate a real-time performance of the MIL-STD-1553B communication with the result of Table 5, the 2ms period is schematically shown in Figure 17. In Figure 17(b), it can be seen that all the periods occurring in the scheduling algorithm with a period of 2ms of the RTiK+ for the MIL-STD-1553B are between 1.993ms and 2.031ms.

	Period of Scheduling algorithm with RTiK+ for MIL-STD-1553B		
	Min	Max	Tolerance
2ms	1.993	2.031	1.57%

(a) Result of Period of 2ms for MIL-STD-1553B



(b) Graph of Period of 2ms for MIL-STD-1553B

Figure 17: A Performance of 2ms Period for MIL-STD-1553B

For a performance measurement of a real-time system, 32 words of data are transmitted according to the period provided by the scheduling algorithm with the RTiK+ in Windows for the MIL-STD-1553B communication, and the miss rate is applied to verify the jobs performed after the deadline. The figure 18 is shown for miss rate during the MIL-STD-1553B communication to decide success or failure in real-time system.

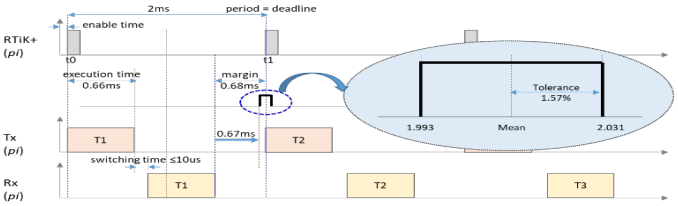


Figure 18: A Miss Rate of the MIL-STD-1553B Communication

The system based on x86 with Windows 8 applied in this paper is a hard-real-time system that must meet a deadline. Therefore, if all deadlines are met, the MIL-STD-1553B communication is a success, otherwise it is a failure.

$$\text{Margin of MIL-STD-1553B} = (pi\text{-}ei)\text{-Tolerance (\%)} \quad (5)$$

Substituting the measured result into Equation (5), actually the margin of the MIL-STD-1553B is 0.39ms that (2ms-1.60ms) - (2ms-1.9938ms). And the miss rate of the MIL-STD-1553B is defined as in equation (6).

Miss rate of MIL-STD-1553B = Number of Error / Number of Communications (6)

The MIL-STD-1553B using PASS 3200 analyzed all data as shown in Figure 14 in the period of 2ms, there was no communication error and all data were normal during the MIL-STD-1553B.

Table 6: Performance of Scheduling Algorithm by RTiK+ for MIL-STD-1553B

Period	Min	Max	Miss Rate
2ms	1.997ms	2.006ms	0%
5ms	4.961ms	5.038ms	0%
10ms	9.995ms	10.007ms	0%

If the MIL-STD-1553B communication period is 1ms and the algorithm proposed in this paper is applied, the maximum transmitted data is Tx(ei) = Rx(ei)) = 0.80ms. This is 0.60ms from the 1ms period of MIL-STD-1553B communication, so even if a double buffer is used, overhead may occur. RTiK+ itself can provide a period within 3% error when there are no workloads on the CPU in a 1ms period.

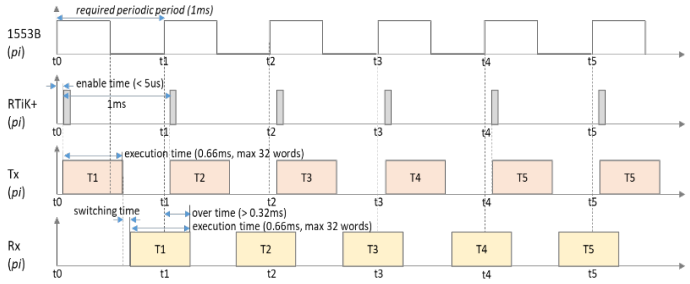


Figure 19: An Analysis Result of 1ms Period for Scheduling Algorithm

It is predicted that MIL-STD-1553B communication will not operate normally due to miss rate because of failure to comply with the deadline due to the lengthy duration as shown in Figure 19.

6. Conclusion

In recent year, with the development of high-speed communication technology and computer technology, the scope of a real-time information sharing has been expanded, and the performance of electronic equipment has been improved remarkably. The accuracy of data collection and real-time data communication are required in industry and defense system.

The RTiK+ presented in this paper controls the MSR_FSB_FREQ register to support a real-time performance on Windows in an environment where x86-based Windows 8 is installed, and calculates the CPU clock tick value required for the operation of the local APIC timer to be a Windows independent timer interrupt was made to occur, and in this way, the matter that local APIC is initialized after booting in Windows 8 is solved.

The experimental verification was checked by measuring the period by calculating a timestamp in the PC's internal memory using RDTSC, and measuring a real-time processing performance of the RTiK+.

And to support a real-time performance in the MIL-STD-1553B communication, The RTiK+ was implanted in the Windows based on x86 system, and a real-time processing functions were added to the BC and RT functions. Here, the real-time performance was provided when BC and RT function of the MIL-STD-1553B communication are implemented, and by applying the scheduling algorithm with RTiK+ for a high-speed MIL-STD-1553B communication such as 2ms of period is implemented to prevent data loss due to error of the period. To proof this scheduling algorithm experimentally, The RTiK+ was implanted in the Window 8 and the MIL-STD-1553B communication of the period of 2ms, 5ms, and 10ms was set to verify whether a real-time processing function of the RTiK+ normally while MIL-STD-1553B communication is performing.

In addition, in order to compare a real-time performance between the RTiK+ and the RTX as a third party for the MIL-STD-1553B communication, the experiment result for the period of the MIL-STD-1553B by applying the experimental environment and experimental method is from 1.993ms to 2.031ms in the period of 2ms. The RTiK+ had a tolerance of 1.5%, and the results of this experiment proved that RTiK+ can be applied to systems by replacing third party such as RTX. Finally, a real-time processing performance of the scheduling algorithm with RTiK+ proposed in this paper for MIL-STD-1553B implemented has a miss rate of 0 during the MIL-STD-1553B communication with a period of 2ms for system that requires high reliability. It also proved that the real-time performance and data integrity for scheduling algorithm with RTiK+ are guaranteed.

In the future research, the methods proposed in this paper is needed in the RS-232C, RS-422 and Ethernet used mainly in industry and defense system, and the research for the Windows 10 based on x86 system is needed as well.

Acknowledgment

I would like to thank my dissertation advisor Professor Lee Cheol-Hoon from CNU for his supervise and expert advice, as well as Mr. Muammar Abdulla Abushehab for extraordinary support.

References

[1] C.M. Krishna, Gang G. Shin, Real-Time Systems, McGrow-Hill, 1997.
[2] C. Koo et al, "Distributed Simulator design by using of SimNetwork to overcome speed limit on GenSim", Recent Advances in Space Technologies (RAST), Proceedings of 5th International Conference on RAST 2011, 430-435, 2011, doi: 10.1109/RAST.2011.5966872.
[3] H. Kim et al., "The Design and Performance Verification of Real-Time Inspection Equipment Software based on WindowsOperating System", Journal of the Korea Contents Association, **17**(10), 1-8, 2018, doi: 10.5392/JKCA.2017.17.10.001.
[4] S. Koh et al., "The method of development for enhancing reliability of missile assembly test set", The Journel of The Korea Academia-Industrial Cooperation Society, **19**(8), 37-43, 2018, doi: 10.5762/KAIS.2018.19.8.37.
[5] J. Kim et al., "Real-time Processing Method for Windows OS Using MSR_FSB_FREQ Control", Journal of Korea Multimedia Society, **24**(1), 95-105, 2021, doi: 10.9717/KMMS.2020.24.1.095.
[6] J. Lee et al., "Timer Implementation and Performance Measurement for Providing Real-time Performance to Windows 10", Journal of the Korea Contents Association, **20**(10), 14-24, 2020, doi: 10.5392/JKCA.2020.20.10.014.
[7] S. Kwon , "Design and Implementation of Air Vehicle Test Equipment for Unmanned Aerial Vehicle", The Journal of Advanced Navigation Technology", **24**(5), 251-260, 2020, doi: 10.12673/JANT.2020.24.4.251.
[8] J. Jung et al., "Analysis for Next-generation High-Speed MIL-STD-1553B Bus Technology", Journal of Aerospace Ssytem Engineering, **11**(6), 76-83, 2017, doi: 10.20910/JASE.2017.11.6.76.
[9] http://www.intervalzero.com
[10] Intel, Intel® 64 and IA-32 Architectures Software Developer's Manual, **2**

(2A, 2B, 2C & 2D), Volume 3A and Volume 3B, Intel, 2016.
[11] Intel, Intel® 64 Architectures x2APIC Specifiation, Intel, 2010.
[12] Intel, Intel Core i7-600, i5-500, i5-400 and i3-300 Mobile Processor Series 2010.
[13] A. Jo, "Integrated Middleware for Real-Time Device Driver on Windows", Journal of Korea Contents Association, **13**(3), 22-31, 2013, doi: 10.5392/JKCA.2013.13.03.022.
[14] H. Lihua, Analysis of Fuel Cell Generation System Application, Ph. D Thesis, Chongqing University, 2005.
[15] https://en.wikipedia.org/Rate-monotonic_scheduling
[16] J. Liu, "Real-Time Systems", Prentice Hall, 2000.
[17] M. Gong et al., "Dynamic Voltage Scaling for Scheduling Mixed Real-Time Tasks", Proceeding ICESS 2007 Third International Conference, 488-497, doi: 10.1007/978-3-540-72685-2.
[18] M. Lee, C. Lee, "Low Power EccEDF Algorithm for Real-Time Operating Systems", The Journal of the Korea Contents Association, **15**(1), 31-43, 2015, uci: 1410-ECN-0101-2016-004-001099668.
[19] M. Jung et al., "Optimal RM scheduling for simply periodic tasks on uniform multiprocessors", Proceedings of the 2009 International Conference on Hybrid Information Technology, 383-389, 2009, doi: 10.1145/1644993.1645064.
[20] M.E. Russinovich, "Windows Internals, Part 2 (6th Edition)", Microsoft, 2012.
[21] https://en.wikipedia.org/wiki/MIL-STD-1553
[22] ILC DDC, MIL-STD-1553 Designer's Guide (5th Edition), ILC Data Device Corporation, 1998.
[23] J. Kim et al., "Real-Time Supports on Tablet PC Platforms", Proceeding IEEE 2020 15th International Conference for Internet Technology and Secured Transactions(ICITST), 111-117, 2021, doi: 10.23919/ICITST51030.2020.935.1322.
[24] J. Kim, "Real-time Processing Method for Windows OS Using MSR_FSB_FREQ Control", Journal of Korea Multimedia Society, **24**(1), 95-105, 2021.
[25] D.A. Solomon, "Inside Microsoft Windows 2000, Third Edition", Microsoft, 2000.
[26] http://msdn.microsoft.com/en-us/library/ms810029.aspx
[27] D.A. Godse, A.P. Godse, Microprocessors, Technical Publications Pune, 432-472, 2007.
[28] O. Bailey, Embedded systems: desktop integration, Wordware Publishing Inc, 34-56, 2005.
[29] Intel, "MultiProcessor Specification (Version 1.4)", Intel, 1997.
[30] J. Choi, "A Study on Developing the Missile System Test Set Software with DDS", Proceeding of Korea Institute of Communication Sciences Conference, **2018**(11), 614-615, 2018, uci: 1410-ECN-0101-2019-567-000045708.
[31] https://en.wikipedia.org/wiki/Front-side_bus
[32] https://en.wikipedia.org/wiki/CPU_multiplier
[33] https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-mmmapiospace
[34] https://docs.microsoft.com/en-us/windows/win32/procthread/scheduling-priorities
[35] http://msdn.microsoft.com/en-us/library/ms685100 (VS.85).aspx
[36] C. Lee, Real-Time Systems, CNU, 2012.
[37] K. Kim et al., "An Indirect Data Transfer Technique based on MIL-STD-1553B for the Software Upgrade of Embedded Equipments on a Missile Assembly with Booster", Journal of the Korea Institute of Military Science and Technology, **19**(1), 43-49, 2016, doi: 10.9766/KIMST.2016.19.1.043.