The Design and Implementation of RTiK+ to Support a Real-time Processing on Tablet PC Platforms

Jong-Jin Kim[1], Sang-Gil Lee[2], Cheol-Hoon Lee[2]
[1]*Tawazun Technology and Innovation, UAE*
[2]*Chungnam National University, Korea*

Abstract

In the case of laptops and tablet PCs that replace desktop, it uses the Windows operating system to support various functions and applications depending on operating system dependency, the Windows operating system does not support real-time processing because it uses multilevel feedback queue scheduling that extends round-robin scheduling. In addition, since the initial value of Local APIC Counter cannot be obtained from the Windows 8, the real-time processing function provided through the existing RTiK no longer work. Therefore, In order to solve this issue, it calculates Local APIC Counter value by using Local APIC and MSR_FSB_FREQ register to support real-time processing function on tablet PCs in this paper. It designed and implemented RTiK+ based on RTiK and RTiK-MP that provides real-time processing function to guarantee the periodicity by calculating the operation time of accurate timer. In order to verify and prove the performance of the RTiK+ implemented on Windows 8, the period was measured using the Read Time-Stamp Counter instruction. Moreover, it was confirmed that it operates to support a real-time processing normally at 1ms and up to 0.1ms period.

Keywords: Table PC, RTiK+, Windows 8, Real Time Operation System

1. Introduction

Recently, with the development of hardware, various mobile equipment used in everyday life are used instead of exclusive equipment for special purposes depending on user environment, and studies for its are being conducted to provide real-time processing functions to mobile terminals such as tablet PCs. In addition, due to the increased performance of mobile terminals, it provides computing performance for interfaces and applications close to personal computer and high portability. For this purpose, the mobile terminals are required to support a real-time processing function with high accuracy for real-time performance of the target equipment and accurate execution of command data transmission [1][2]. In the case of this real-time processing function, the research has been actively conducted in fields such as control of unmanned aerial vehicle, inspection of weapon systems and medical equipment requiring high accuracy. For example, in the case of the United States, which is a military power, a various technologies development and evaluation of performance using Tablet PCs are underway, such as developing a detection technology that transmits video clip on unmanned airplanes to tablet PCs in real-time [3][4].

However, in the case of a laptop and a tablet PCs that replaces a PCs, the Windows as an operating system is operated to provide compatibility with existing developed libraries, dependency on the operating system of the operating environment, and various expansion functions. In the case of the Windows operating system, there is a problem that the real-time processing performance for high priority is not always provided due to the uniform processing of various processes and threads.

Therefore, commercial solutions must be utilized to support capability of a real-time processing on tablet PCs to end user or developer. The Real-Time Extension (RTX) from IntervalZero Company and INtime from TenAsys Company have already been researching to support function of a real-time processing at the Windows 8 to provide that function for 64-bit Windows operating system such as the Windows 8. However, it also brings to increase the costs of development due to the highly unit price for purchase, royalty after porting and the cost of maintenance per year. In addition, in the case of RTiK (Real-Time implanted Kernel) and RTiK-MP (Real-Time implanted Kernel for Multi-Processor) released by CNU for the substitution of RTX applied in Korean weapon systems [5][6][7] in order to provide a real-time processing function to the Windows operating system. Since the initial value of The Local APIC Timer used for the independent timer from the Widows is reset to “0” for the internal protection when starting the operating system from the Window 8, there is a problem that the method of calculating the period time cannot be utilized through the existing Local APIC Counter value.

In this paper, in order to solve this problem, it implements a real-time processing function on the tablet PC applied the Windows 8 to provide a real-time processing function. Based on the Local APIC Counter, it improved the period part to control the

MSR_FSB_FREQ register that determines the CPU operating frequency. It improved the part counting the period to control the MSR_FSB_FREQ register, which determines the CPU operating frequency, a timer counter value that operates differently for each CPU is determined. It designed and implemented the RTiK+ to support a real-time processing function on the Windows 8 operating system through effective power management by dynamically controlling C-State to access a CPU power management technology provided by Intel Corporation. The RTiK is implanted into the tablet PC and then a real-time processing function is verified by proving that it guarantees accurate time of performance for 1ms and 0.1ms period. In this paper, the study to support the function of a real-time processing is being processed for the Windows 8 on the tablet PC to solve aforementioned problem.

This paper describes the RTiK and RTiK-MP which supports function of a real-time processing and low-power technique on the Windows 8 as a related research in section 2, the design and implementation of RTiK+ to provide the function of a real-time processing on the tablet PCs installed with the Windows 8 in section 3, the experimental environment and results in section 4, Actual application examples in section 5, and the conclusion and future research theme in section 6.

2. Related Research

Describes the RTX and INtime, third party operating systems currently in use, describes also the RTiK and RTiK-MP, which was released to provide function of a real-time processing on the Windows operating system.

2.1. RTX

IntervalZero developed RTX operating in Kernel mode (Ring 0) in 1995 based on x86 PC as shown in Figure 1.

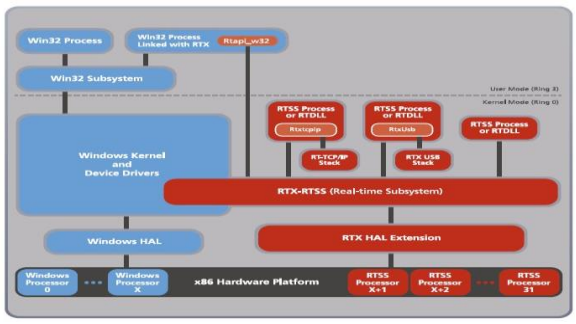

Figure 1. RTX Architecture

RTX is extension software that adds the function of real-time operating system to be able to control high performance a real-time to the Windows 8. It provides to use the advantages of existing real-time operating systems using the RTX to most developers in a familiar and convenient Windows environment. In other words, the RTX is not pure real-time operation system, but software that supports for limitations such as real-time performance by taking full advantage of the popularity of Windows and affluent GUI libraries. It is utilized in many business fields such as defense, aerospace, instrumentation, simulation, medical and robotics [8].

2.2. INtime

TenAsys developed INtime operating in User mode (Ring 3) in 1997 as shown in Figure 2.

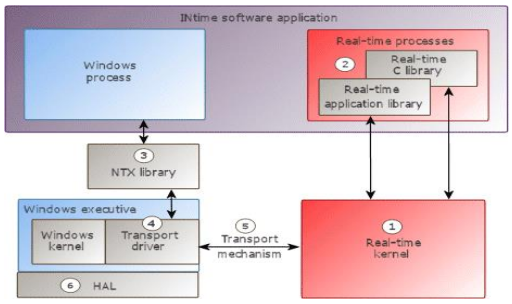

Figure 2. INtime Architecture

INtime is a real-time system software specified in x86 CPU that uses the iRMX kernel developed by Intel corporation. It uses the main processing on the system that installs the INtime software in the control and measurement systems using Windows XP, Windows 2000 and Windows 8. It can support real-time performance of high reliability and precision by maintaining the flexibility of the Windows. The INtime is a real-time operating system that operates in cooperation with Windows XP, Windows 2000 and Windows 8 as an industrial computer, x86 based on computer and general computer and runs simultaneously with Windows on the same hardware. Windows installed with INtime is a multi-kernel system running two kernels. Therefore, it operates in the type of using two virtual CPUs [9]. Table 1 shows major difference of RTX and INtime.

Table 1. Third party operating systems on the Windows

Classification	RTX	INtime
Development Company	IntervalZero	TenAsys
Kernel Conversion	Software-like kernel conversion	Hardware-like kernel conversion
Operation Mode	Kernel Mode (Ring 0) operation	User Mode (Ring 3) operation

2.3. RTiK

2.4. RTiK-MP

RTiK-MP, extended version of RTiK, to support multi-processor was developed by CNU since 2013. RTiK-MP provides the function of a real-time processing on the Windows of x86 based on multicore environment as shown in Figure 4. It is ported in the type of device driver and able to utilize the kernel resources and hardware resources of the Windows.

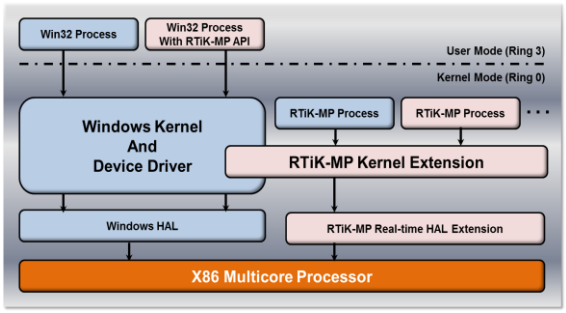

Figure 4. Configuration of multi-core based on real-time Windows operating system

The RTiK-MP accesses and controls x86 multicore hardware in the real-time HAL extension, and creates and manages a real-time thread using API provided by the RTiK-MP in the Windows kernel. The RTiK-MP is being used in the test equipment for weapon systems so far. In addition, the research for the RTiK-MP has been actively conducted to develop communication middleware to provide the function of a real-time processing in various fields environment.

The RTiK-MP generates a windows-independent timer interrupt by using a local APIC to provide a capability of a real-time processing on the Windows based on PC environment.

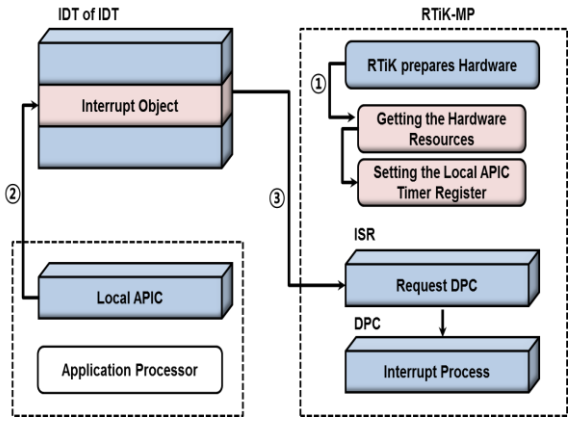

Figure 5. Interrupt process of RTiK-MP

As shown in Figure 5, the RTiK-MP guarantees periodical operation of a real-time thread using a local APIC of Application Processor (AP) instead of Boot Strap Processor (BSP) to generate a windows independent timer interrupt. When a local APIC timer interrupt creates, the interrupt handler registered in the Interrupt Descriptor Table (IDT) through a vector serving as an index branches in the IDT of the AP. The handler executed at this time is registered in the queue for DPC to minimize the interrupt latency.

When the interrupt handler is completed and if IRQL value becomes dispatch level, the interrupt handler executes the registered handlers in the dispatcher. Through this process, the RTiK-MP provides the function of a real-time processing on the Windows which is a general-purpose operating system [5][6][7].

2.5. Local APIC timer

APIC is an interrupt controller provided by the x86 architecture. In other words, it implements the function of transferring the hardware interrupt to the IDT that has the address of the interrupt handler. APIC is composed of Local APIC and I/O IPIC. I/O APIC receives IRQ signals from external devices and transfers them to Local APIC that is transferred to the CPU by Local APIC to handle interrupts.

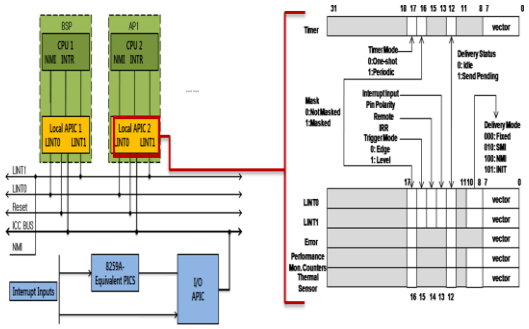

Figure 6. Operation of Local APIC timer

Local APIC is an interrupt controller provided by the x86 architecture as an extension of PIC. APIC operates a role of delivering hardware interrupt to IDT that has the address of the interrupt handler. A single core and multicore have different processing for Local APIC in a CPU. Local APIC based on multicore exists for each core, and interrupts can be generated in each Local APIC. Accordingly, an IDT accessed when an interrupt occurs is also present for each core. Therefore, it has to be designed and implemented to generate a timer interrupt by selecting one of the two cores using the Local APIC of the corresponding core. In the case of RTiK-MP, a timer interrupt is generated by selecting an AP among processors and using the AP's Local APIC. Figure 6 shows the structure of AP's Local APIC when there are two cores. As shown in Figure 6, RTiK-MP guarantees periodical operation by generating a timer interrupt independent of Windows by using the timer-related registers of the AP's Local APIC [11][12].

2.6. Deferred Procedure Call (DPC)

DPC means that the ISR of the hardware interrupt uses a CPU only for important tasks. This is the Windows mechanism provided to do so [13]. IRQL stands for Interrupt Request Level (IRQL), and a code executed in a given IRQL cannot be interrupted by an IRQL code that is lower or equal to that. Table 2 shows each IRQL and its description.

Table 2. Interrupt Request Level (IRQL)

IRQL	Name	Description
31	High	Bus error, address error, reset, ⋯
30	Power	Power failure
29	Inter CPU	Inter processing notification
28	Clock	Real time clock
27	Profile	Performance measurements
26	Device n	I/O Controllers
⋯	⋯	⋯
3	Device 1	I/O Controllers
2	Dispatch	Thread dispatching and DPC interrupts
1	APC	Asynchronous procedure calls
0	Passive	All interrupts enable

As shown in Figure 7, when an interrupt occurs, the IRQL is raised to execute the ISR. If the thread to be executed is less important, request a DPC and change the IRQL at that time the ISR is terminated, and when the operation level of the DPC is lowered to the dispatch level, execute the processing function of the DPC and then return to the original thread.

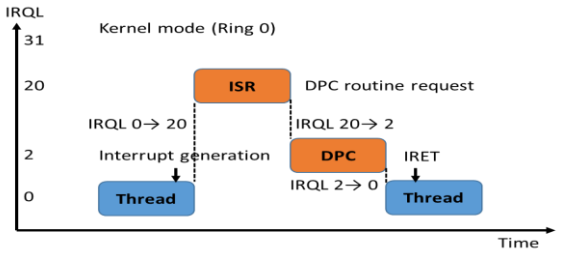

Figure 7. Change of IRQL during interrupt processing

In this way, when the ISR of the hardware interrupt uses a CPU, a mechanism provided to use the CPU only for more important tasks and to resume the remaining tasks at a lower priority IRQL is called a DPC.

2.7. Power control in the CPU

As the performance of a mobile devices has improved and functions have become more various. Therefore, the importance of power management to support these functions is highlighted. The low-power technique is highlighting importance in the mobile processor which uses time of battery is important by a technique to reduce the power consumption as controlling the operation state of a CPU depending on the use rate of the CPU for a certain period of time and the platform. Normally, for mobile devices in the x86 architectures of embedded system, the low-power technique provides five status of the CPU called the C-State as shown in Table 3.

Table 3. Power status of the CPU

C-State status	Description
C0	Able status of the CPU
C1	Idle status of the CPU
C1E	Idle status due to internal clock of the CPU stop and the voltage decrease
C3	Stop status for all internal/external clocks of the CPU and disable of L1/L2 cache
C6	Sleep status that protects the voltage of the CPU after the status of the CPU is saved

As shown in Table 3, the C0 is a status in which the CPU operates the processes. The C1 and the C1E make idle status of the CPU to decrease the clock and the voltage, thereby reducing the power consumption. The C3 and the C6 stop the operation of the CPU after the operation of the L1/L2 cache stop, and the low power is provided. It is to reduce the power consumption depending on the status of the C-State in Windows. The change of the CPU's clock affects that delay time for processing the Local APIC timer interrupt occurs. However, there is a problem that affects the most important determination of time in the real-time system due to synchronization. Therefore, it is necessary to study the efficient control of the C-State to guarantee the determination of time [14].

2.8. MIL-STD-1553B communication

MIL-STD-1553B, which is mainly applied as an aircraft weapon system and a guided missile weapon system, is a dual-redundant and bi-directional serial communication method with a communication speed of 1Mbps. MIL-STD-1553B communication bus is shown in Figure 8. As shown in Figure 8, a communication transmits messages using a prescribed communication protocol between different types of electronic equipment in an aircraft. It is a communication standard mainly used for military and aerospace, and it is used for equipment requiring high reliability in weapon systems.

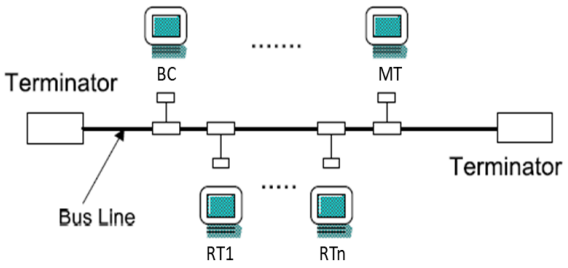

Figure 8. Block diagram of MIL-STD-1553B communication interface

It is composed of BC (Bus Controller), RT (Remote Terminal) and MT (Monitoring). BC controls the data bus and RT is controlled by BC to transmit or receive data and MT functions to monitor all data on the bus [15].

3. Design for a Real-Time Processing on the Tablet PC

In case of existing the RTiK-MP developed to provide a real-time processing function to the Windows operating software, it uses Local APIC timer register provided by Intel's CPU. At this case, it was possible to calculate the reference time using the Local APIC based on counter register. However, since Windows 8, Local APIC register value is initialized to "0" after booting the operating system. Thus, there is a problem in providing a real-time processing function corresponding to the system clock. In addition, the C-State mode is used to reduce power consumption in mobile terminal to increase battery's operating time. The function reduces the power consumption by adjusting the clock by automatically changing between five states depending on the load of a CPU. However, this function has a problem that the system clock is changed dynamically, and the accurate period calculation is not applied.

In order to solve this problem in this paper, it designed the RTiK+ that improved the RTiK that provides a real-time processing function to x86 based on the Windows operating system. The RTiK+ is implemented as a type of device driver and it has a hardware abstract layer (HAL) for hardware access and controls a real-time timer value and the C-State through the MSR_FSB_FREQ register. It performs a real-time task management through a period occurring in a timer of an accurate period.

3.1. A real-time processing utilizing local APIC Time

The RTiK+ creates a windows-independent time interrupt through the RTiK+ access kernel resources and processing sequence is shown in Figure 9. As shown Figure 9, the RTiK+ uses local APIC and kernel resources to provide the function of a real-time processing in the Windows 8.

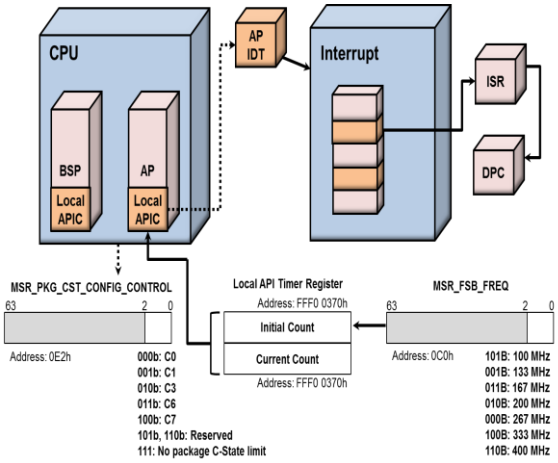

Figure 9. Method to provide a real-time processing by RTiK+

Before the timer interrupt is generated, the initial counter value of the Local APIC of AP is set by the MSR_FSB_FREQ register so that a windows-independent timer interrupt is generated. It is implemented to create a real-time task by registering an interrupt object in the IDT for interrupt processing. In addition, the RTiK+ utilizes the MSR_PKG_CST_CONFIG_CONTROL to support the time determination of the generated real-time task, and it controls the operation mode of the CPU. Therefore, the RTiK+ guarantees the periodicity by minimizing the margin of error range of the set task period [16][17][18].

3.2. Generating timer interrupt through controlling the FSB

The front-side-bus (FSB) is the data path between a CPU and a memory controller hub, and it is the external bus to communicate between a CPU and a peripheral device. In addition, the FSB provides the function that determines the operating speed of a CPU using a multiplier and provides synchronization as it affects the memory clock speed. A block diagram of chipset at the x86 architecture is shown in Figure 10. As shown in the Figure 10, the frequency generated by the clock generator is connected to the FSB and it provided to the CPU so that it is set as the initial counter value of the Local APIC timer register. Then it determines the occurrence time of the timer interrupt.

This means that the local APIC timer generated by the CPU is controlled through the operating frequency of the FSB. In the case of Windows 8, which is a mobile operating system, it provides the operating frequency of FSB depending on the CPU through the lower three bits of the MSR_FSB_FREQ register that is a hardware resource.

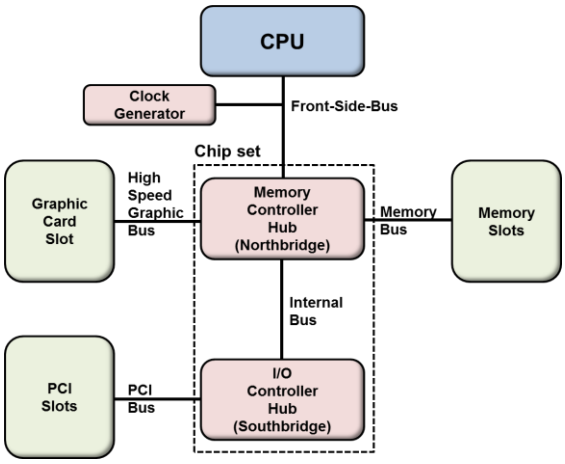

Figure 10. Block diagram of x86 architecture

Therefore, in order to determine the occurrence period of local APIC timer interrupt of the RTiK+, the operating frequency value of the FSB is known through accessing the MSR_FSB_FREQ register and it sets an initial counter value before activating the local APIC timer and generates a windows-independent timer interrupt. When a timer interrupt generates, it branches to the interrupt handler and the performed handler is implemented to call the delay processing function to minimize the delay time [19].

3.3. Calculating the determination of time through controlling the C-State

As Windows 8 decreases the clock of a CPU and turns off the power through the C-State's access for control of the low power of a CPU, it provides the function of real-time saving power that reduces power consumption. The following formula is applied to determine the clock of the CPU in the Windows 8.

CPU Clock = FSB Frequency x Multiplier

Because the FSB has a value specified depending on the hardware specifications of the system, the Windows 8 provides low power by controlling the multiplier value.

However, the method of adjusting the multiplier value to decrease the clock of a CPU has a problem that it is difficult to guarantee determination of time for the Local APIC timer. The variation of the multiplier value frequently changes the processing speed for the local APIC timer of a CPU, and it causes time error with the periodically Local APIC timer interrupt occurred through the initial count value set previously. In addition, there is already mentioned problem because the delay time occurs due to the time for activating the CPU of idle state or sleep state. Therefore, in this paper, it designed control the C-State by accessing the MSR_PKG_CST_CONFIG_ CONTROL register provided by the x86 architecture to guarantee the determination of time

After RTiK+ is implanted on the Windows 8, the clock of a CPU depending on the setting of the C-State using a program called HWM BlackBox is shown in Figure 11. As shown in (a) of Figure 11, when C-State is activated, it can be check that the clock is controlled depending on the CPU's utilization rate. As shown in (b) of Figure 11, when C-State is deactivated, it keeps the maximum value of clock without the CPU's utilization rate.

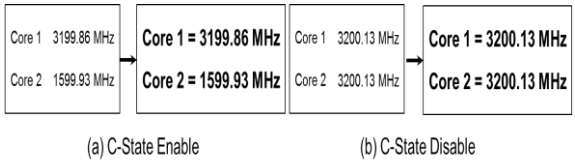

Figure 11. Change of the CPU clock depending on the setting of the C-State

The RTiK+ accessed to the MSR_PKG_CST_ CON_FIG_CONTROL register provided by the x86 architecture with MmMapIoSpace(), the kernel API, before setting the timer interrupt. And the processing speed of the CPU is maintained by setting the value of three bits of the lower to C0 (000B) to guarantee the determination of time for the RTiK+ as shown in Figure 12.

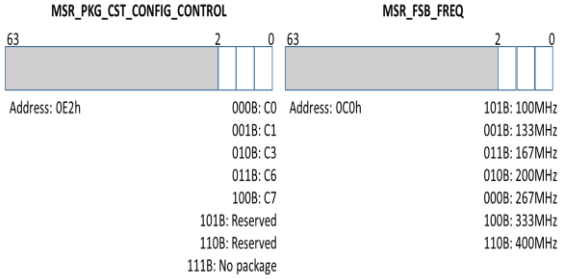

Figure 12. Access to registers for setting

In addition, if the RTiK+ is uninstalled on the Windows 8 or it is disabled, it is designed to minimize the influence on the Windows as implementing to be changed to the previous state.

3.4. Providing a real-time processing function in user area of RTiK+

In the case of the existing RTiK, it is implemented as a type of device driver to directly access the kernel resources such as the local APIC timer of the CPU. In addition, In the case of RTiK-MP, this makes that it easy to access hardware resources in the kernel area, but there is a problem that the system may be adversely affected when users and developers incorrectly access to the kernel resources. In addition, the research for a real-time processing function of the

user area that has been studied occurs the loss of event signal in the multicore environment, so this has a problem that the period, accuracy of the data and integrity of the data are not guaranteed. Therefore, the RTiK+ is designed to guarantee a real-time processing of thread in user area to guarantee the accuracy and integrity of the data. Figure 13 shows operation process of the RTiK+ to provide the function of a real-time processing in the user area. User creates a real-time thread with API provided by the RTiK+. It guaranteed a real-time processing of thread as the created real-time thread by receiving periodic signal transferred from the kernel area runs the code.

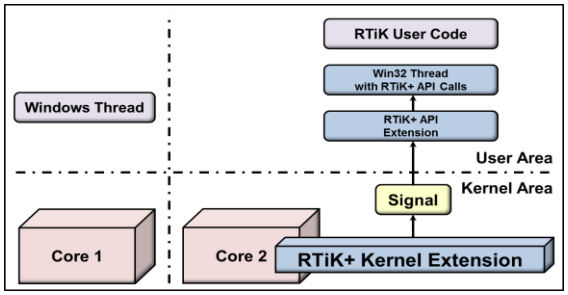

Figure 13. Operation process in the user area of the RTiK+

In case of windows' environment with multi-core, the loss of event signal occurs because event occurrence is notified only to processes operating in the corresponding core.

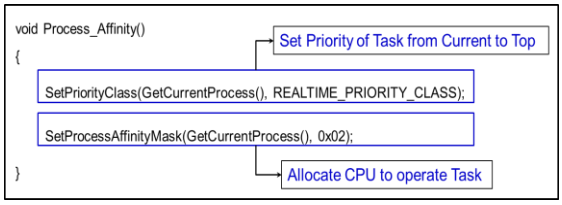

Figure 14. Setting the core to run real-time thread

In order to solve this problem, the generated real-time threads are only operated in the core that RTiK+ is implanted and it prevents the loss of event signal. The implemented code into the RTiK+ to guarantee the accuracy and integrity of the data is shown in Figure 14. As shown in Figure 14, it is implemented to call the Process_Affinity() function in the real-time thread and the real-time thread is operated only into the core in which the RTiK+ operates through the SetProcessAffinityMask() function as Windows' API in internal thread. In addition, by setting the priority for real-time thread of the RTiK+ and the priority for the process to the highest level, it is not to preempt the CPU from other windows threads. The code is implemented into the RTiK+ to provide the function of a real-time processing to user area as shown in Figure 15.

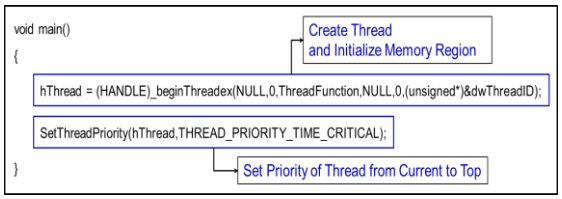

Figure 15. Setting the Priority of real-time thread

As shown in Figure 15, when the real-time thread is created, it is set to THREAD_PRIORITY_ TIME_CRITICAL as the highest thread priority as Table 4 that is provided in the Windows and is set to REALTIME_PRIORITY_CLASS as highest task and Table 5 [20]. It implemented more operate than other threads of windows.

Table 4. Property of thread priority

Property	Priority level of threads
7	THREAD_PRIORITY_TIME_CRITICAL
6	THREAD_PRIORITY_HIGHEST
5	THREAD_PRIORITY_ABOVE_NORMAL
4	THREAD_PRIORITY_NORMAL
3	THREAD_PRIORITY_BELOW_NORMAL
2	THREAD_PRIORITY_LOWEST
1	THREAD_PRIORITY_IDLE

In other words, as shown in Table 4, the priority of windows threads is changed to THREAD_ PRIORITY_TIME_CRITICAL as the highest priority provided by Windows so that the priority was set higher than other threads running in the Window 8.

Table 5. Property of process priority

Property	Priority level of threads
6	REALTIME_PRIORITY_CLASS
5	HIGHT_PRIORITY_CLASS
4	ABOVE_NORMAL_PRIORITY_CLASS
3	NORMAL_PRIORITY_CLASS
2	BELOW_NORMAL_PRIORITY_CLASS
1	IDLE_PRIORITY_CLASS

In addition, as shown in Table 5, windows process can have six priority groups and the priority of the created process is determined as one of them. When the RTiK+ process is created, it has to create a process by setting it to the REALTIME_PRIORITY_CLASS priority level that is the highest priority. The REALTIME_PRIORITY_CLASS priority has a thread priority level of 31 to 16. The runtime priority in Windows change occurs only from 15 to 0, so that the CPU is not preempted by other threads.

4. Experimentation Environment and Results

An experimental environment was configured to measure the real-time performance of RTiK+ implanted on the Windows 8 operating software through DPC and FSB control in the Windows as shown in Figure 16. RTiK+ was ported on the host computer and the target computer monitored it.

4.1. Experimentation environment and method

The RTiK+ was successfully ported on the Windows 8 operating system of host computer and the experimental environment for verifying the capability of a real-time processing function is configured as shown in Figure 16 with specifications as shown Table 6. The RTiK+ was ported to the host computer that is a tablet PC, and although it can be run alone without other system, but it has been verified and measured through connection with a monitoring computer connected via USB-to-serial cable for debugging for development and measuring for the verification of performance.

Tablet PC platforms were used as host computer and desktop computer was used as target computer to provide different experimental environment using different CPU and OS. Moreover, target computer was interfaced with measurement instrument such as oscilloscope for measuring data.

In the experimental method, a real-time thread with a period of 1ms was created in user area, and the time for each period was measured. At this time, the thread of the host computer implanted RTiK+ was operated to check the real-time performance.

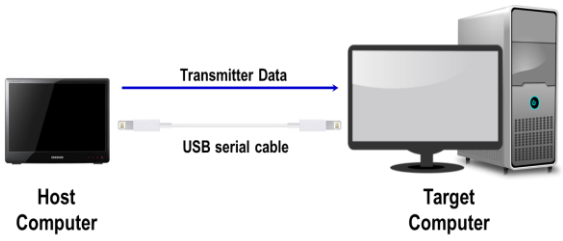

Figure 16. Environment of experimentation for providing the RTiK+ operation

Table 6. Configuration of the host and target computer

Item	Host Computer	Target Computer
CPU	Intel Pentium Dual-Core 2117U @1.86 Ghz	Intel Core i5-2500 @3.30 Ghz
Operating System (OS)	Windows 8	Windows 7
Development Software	Visual Studio 2008	Visual Studio 2008

In addition, in order to measure the time when there are other threads running in the Windows 8, a process that executes an infinite loop of while function in C language was created to measure each period. For the accuracy of the experimental results, signals were output to the I/O ports every period, and it were measured by an oscilloscope. Then it checked the results for 1ms period of oscilloscope as shown in Figure 17. However, method of following Figure 18 mainly was utilized for measuring performance of real-time processing because of saving many data for periodical accuracy during experimentation.

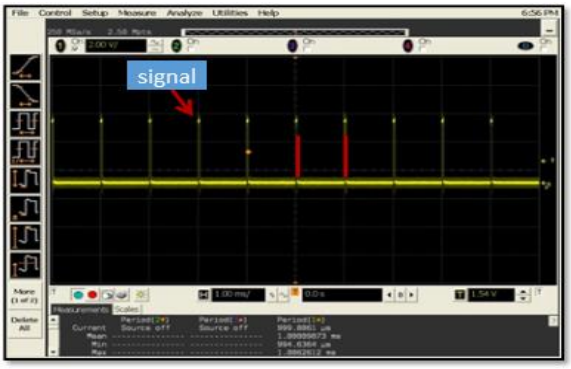

Figure 17. Signal's output for measuring 1ms period with oscilloscope

In addition, it measured the period using the command of Read Time Stamp Count (RDTSC) that returns the number of clock tick to prove the performance of the RTiK+ in this paper. The implemented code with RDTSC is shown in a real-time thread for the experimentation in Figure 18.

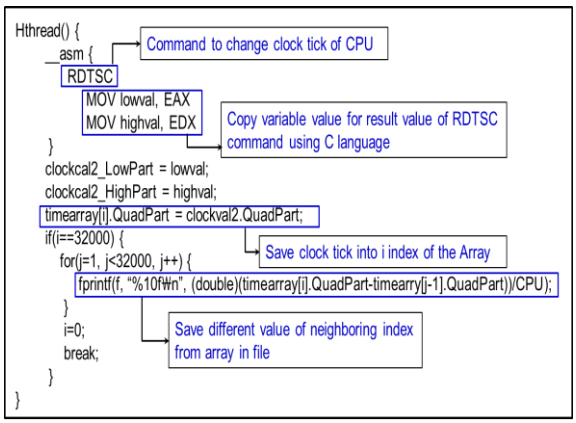

Figure 18. Implemented test code for measuring the period of the RTiK+

As shown in Figure 18, it stored the value of the clock tick in the array whenever the real-time thread of user area is executed after a timer interrupt with a period of 1ms occurs. When the execution of the RTiK+ is completed, it measured the period by the method of calculating the difference of the clock tick stored. In addition, to prove that the determination of time is satisfied with the RTiK+, the measurement is performed for the period depending on the status of the C-State in the same experimental condition and environment. In order to verify that the real-time thread preempts and operates the CPU without affecting other threads of the Windows 8, it created workloads performing repeatedly like while function in C language and measured the period.

4.2. Experimentation results

4.2.1. Performance measurement for a real-time processing of RTiK+. Period of 1 millisecond and 0.1 millisecond was measured to prove performance of a real-time processing using RTiK+. Because UAV system and weapon systems sometimes use fast communication such as less than 3ms to control actuator and engine.

A. Measurement of 1ms period: After making the status of the C-State from enable to disable and setting 1 millisecond for the period of the RTiK+, it measured the period of the RTiK+ by operating simultaneously RTiK+ without workloads and RTiK+ with ten workloads as shown Figure 19. The maximum number of the data that can be plotted by the graph in Microsoft excel is 32,000 data points. Therefore, it measured the period of the RTiK+ when the operation of RTiK+ is run up to 32,000 times. The value of x-axis means the number of operations of the RTiK+ for supporting a real-time processing and the value of y-axis means period.

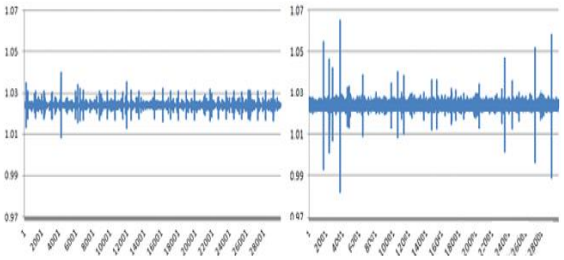

Figure 19. Measurement of period of 1ms for the RTiK+ without workloads (left), and with ten workloads (right)

The maximum and minimum value of results for the period measured is showed in Table 7. The RTiK+ periodically operates that the real-time thread has a margin of error of about 3% when RTiK+ does not have a workload as shown in the Table 7. In addition, The RTiK+ periodically operates that a real-time thread has a margin of error of about 6% when RTiK+ has ten workloads. Therefore, the RTiK+ is not influenced by many parallel processes operated in real time.

Table 7. Measurement of period of 1ms without workloads, and with workloads

Classification	1ms Period without workloads	1ms Period with workloads
Maximum	1.039777 ms	1.062404 ms
Minimum	1.008465 ms	0.957421 ms

B. Measurement of 0.1ms period: Figure 20 is graph showing the periodical performance with higher resolution. The graphs show the results of measuring the period of RTiK+ set to 0.1 millisecond.

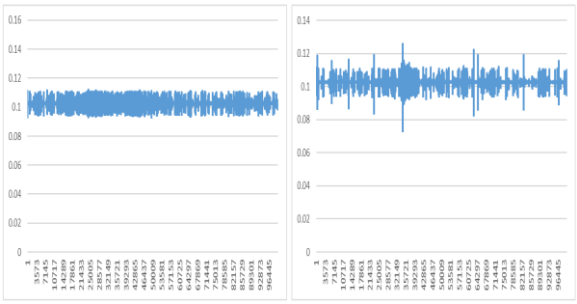

Figure 20. Measurement of period of 0.1ms for the RTiK+ without workloads (left), and with ten workloads (right)

Left graph in Figure 20 shows a real-time performance for a system without workloads, and right graph shows performance for a system with ten workloads. The value on the x-axis represents the number of the RTiK+ operations to support a real-time processing and the value on the y-axis represents the time of the task call period.

Table 8. Measurement of period of 0.1ms without workloads, and with workloads

Classification	0.1ms Period without workloads	0.1ms Period with workloads
Maximum	0.112016 ms	0.125625 ms
Minimum	0.093311 ms	0.735692 ms

Table 8 shows the maximum and minimum values of measurement results. As shown in Table 8, the real-time thread has a margin of error of about 10% when there are no workloads, and it seems that it operates periodically. In addition, when ten workloads are applied, a maximum margin of error of 20% occurs. In the call period of task of 0.1 millisecond that is a more precise period, it seems that the call period of

the task is kept within the allowable error range in the program of the user area.

4.2.2. Performance measurement for multi-period task of RTiK+: After C-State is disable, the results of measurement that simultaneously is executed by setting the task with various period of RTiK+ is shown in Figure 21. In a system requiring real-time performance, the operation of task having various periods can be required. In order to support this request, it runs the testing by operating task with several different periods with same experimental condition. The periods of RTiK+ to provide real-time processing were sequentially set to 3ms, 5ms, 10ms and 25ms. In these experiments, about 30,000 experimental data were measured. The value on the x-axis represents the number of operations and the value on the y-axis represents the period.

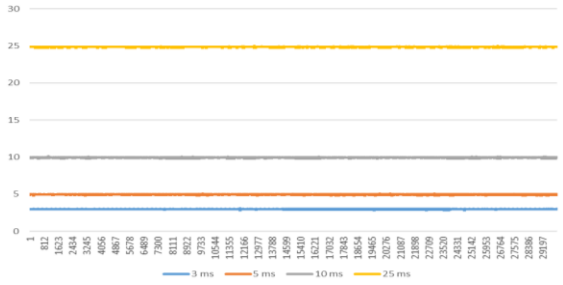

Figure 21. Test result for period of 3ms, 5ms, 10ms and 25ms after C-State disable

As shown in Figure 21, even if tasks with different periods are executed, it seems that they are run normally within the set period.

Table 9. The results after C-State disable

Classification	Tasks with Period of RTiK+			
	3ms Period	5ms Period	10ms Period	25ms Period
Maximum	3.072	5.064	10.114	25.027
Minimum	2.879	4.903	9.840	24.786

Table 9 shows the results of measurement in Figure 21. In each case, it is verified that it operates within a margin of error range of about 2%. This means that the performance of a real-time processing can be stably provided even in an environment where tasks having different periods are simultaneously executed.

4.2.3. Performance measurement for a real-time processing of RTiK+ after C-State enable: When the C-State is enabled to control low power of a CPU. Then a real-time thread with 1 millisecond period of the RTiK+ to provide a real-time processing with same experimental environment and condition is created. The results of testing is shown in Figure 22.

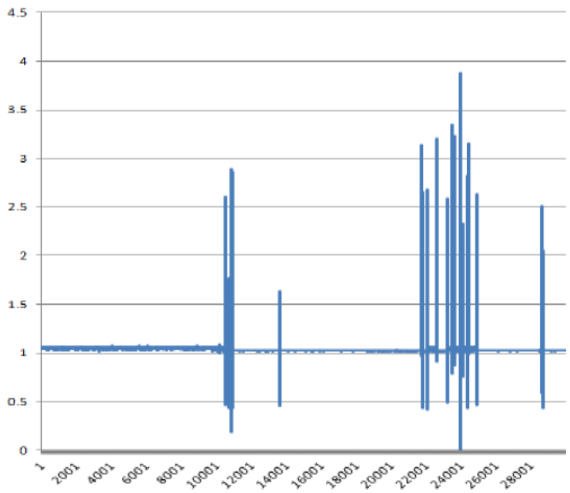

Figure 22. Measurement of period of 1ms after C-State enables

The maximum and minimum of the results of period measured is shown in Table 10. As shown in Table 10, when the C-State is enabled, even if there are no workloads, it seems that it is largely out of the set 1 millisecond period. Therefore, it seems that the enable of C-State means that it does not guarantee the determination of time in the weapon systems or in the test equipment.

Table 10. The results after C-State enables

Classification	Measurement Result of 1ms period after C-State enables
Maximum	3.876373 ms
Minimum	0.001233 ms

4.2.4. Comparison of Performance: The experimentation with same test environment and condition is performed to compare the real-time processing performance between RTiK+ and third party (RTX) that is already released for commercial solution of real-time operating software. It measured the real-time processing performance of RTiK+ and RTX by setting period of task execution to 5ms and 10ms. As shown in Table 11, it was able to confirm that it operates normally without big different period. In addition, it proved that RTiK+ is equivalent to RTX in this paper through the test results for only a real-time processing performance.

Table 11. Comparison with RTiK+ and RTX

Period	RTiK+		RTX	
	Maximum	Minimum	Maximum	Minimum
5 ms	5.064 ms	4.903 ms	5.252 ms	4.999 ms
10 ms	10.114 ms	9.840 ms	10.003 ms	9.996 ms

5. Real World Application

In order to verify the performance of a real-time processing in the actual test equipment of weapons system based on the Windows 8, RTiK+ was applied.

This test equipment in the guided missile weapon system performs the function of checking the electrical functions and algorithm performance of major electronic components such as guidance control unit, inertial navigation unit, seeker and actuator. When it operates the inspection function, each component is controlled in real time through MIL-STD-1553B communication, and the result data is checked in real time. In this case, the function of the real time processing is required essentially in real-time for periodical communication.

As shown in Figure 23, a discrete I/O board, an analog I/O board and a MIL-STD-1553B board are mounted on the Windows operating system based on the industrial PC system. This test equipment is interfaced with each component of the guided missile weapon system.

In order to provide a real-time processing function, as shown in Figure 24, the transmit command and receive command are executed periodically in RTiK+ threads. The communication period is also set to a maximum of 2 milliseconds according to request of system, and the RTiK+ ported in the BC function and RT function is activated to support a real-time performance so that data is transmitted and received from the real hardware of the MIL-STD-1553B board.

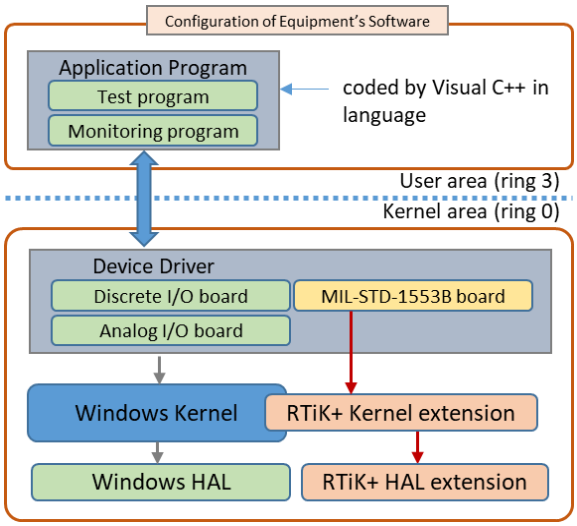

Figure 23. Software configuration of test equipment based on Windows 8 with RTiK+

```
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);   <- Generate transmit command
if (status == FAILURE)
{
    printf("FAILURE\n");
    printf("%s\n", sbs_read_error());
}

uSleep(600);   <- Receive data
m11553_read_link_data(device_number, TxLink.link_id, TxLink.buff_id[0], r_buffer);   <- Check receive data

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_read_link_data(device_number, RxLink.link_id, RxLink.buff_id[0], r_buffer);   <- Set transmit data

status = m1553_load_chain(device_number, 1);
if (status == FAILURE)
{
    printf("FAILURE\n");
    printf("%s\n", sbs_read_error());   <- Generate receiver command
}
uSleep(600);
```

Figure 24. The implemented code to transmit and receive data with MIL-STD-1553B board

In addition, whether data overhead occurs in the buffer while 1Mbps communication is in progress, it was implemented continuously to monitor using the active buffer point provided from the MIL-STD-1553B board, and immediately to notify an operator when an error occurs.

The experimental configuration for measuring the real-time processing performance was composed of host equipment and target equipment in actual test equipment of weapon system as shown in Figure 25.

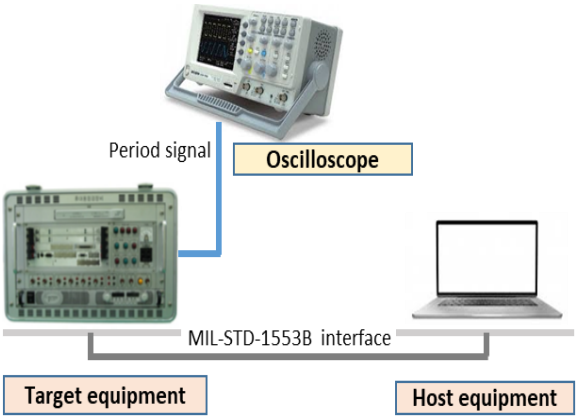

Figure 25. Configuration of experimental environment for measuring performance

In the experimental methodology, MIL-STD-1553B communication was operated in the user area with RTiK+ ported in the test equipment, and a real-time thread having a period of 2 milliseconds was created to measure periodically communication.

Table 12. Measurement of period of 2ms with the RTiK+ in MIL-STD-1553B communication

Classification	2ms Period without workloads	2ms Period with workloads
Maximum	2.0315028 ms	2.0065652 ms
Minimum	1.9938777 ms	1.9978151 ms

As shown in Table 12, a maximum margin of error of 1.5% occurred. It was confirmed that the task call period is maintained within the allowable error range in the user area program in this experimentation.

6. Conclusion and Future Research

In this paper, to support the real-time processing on the tablet PC platforms environment installed the generally Windows operating system such as Windows 8, it improved a problem for the operation of timer interrupt that operates independently Windows operating system by calculating the tick value of the CPU needed for the operation of Local APIC Timer with the MSR_FSB_FREQ register.

The RTiK+ is also designed to reduce the margin of error of period because of the C-State operated to solve a problem of power consumption in the mobile PCs environment. It operates normally at the set period by controlling the operation mode of the CPU with MSR_PKG_CST_CONFIG_CONTROL register to guarantee determination of time in the state that the real-time timer operates. In order to guarantee the accuracy of data and integrity of data in the user area, the real-time thread is implemented to operate only into the corresponding core that the timer interrupt generates, and the function of real-time processing is provided.

In addition, it used the RDTSC to measure the real-time performance of the RTiK+ implemented on Windows 8 based on Table PC. When it executed the minimum period of 1 millisecond of the created real-time thread, it operated with a margin of error of about 3% when there are no workloads and it operated with a margin of error of about 6% when there are workloads. In addition, when the workloads is applied on purpose to simulate real condition, it has been proven that even if the scheduled interrupts frequently occur, the CPU is preempted and operates without being affected by other threads of the Windows.

In addition, in order to provide a more precise period up to 0.1 millisecond period, it was verified that it operates with a margin of error of about 10% when there are no workloads and about 20% when the workloads for 0.1ms period. In this paper through this experiment, it was proven that the real-time processing performance of RTiK+ on the Windows 8 is guaranteed even in the faster system that the real-time processing function is needed.

In the actual weapon system in use, it also was confirmed that the function of a real-time processing for MIL-STD-1553B communication from RTiK+ applied in the test equipment based on the Windows 8 normally operate to provide real-time performance.

In the research theme of future, in order to verify the high reliability of the RTiK+, the real testing on the test equipment used and required in the actual weapon systems is needed and it is necessary to implement an API for developers easily to utilize the RTiK+ more conveniently. In addition, in order to support the performance of a real-time processing on a variety of equipment, the research for extension function to apply the real-time application is needed based on the various Windows operating software.

7. References

[1] C.M. Krishna, G. Shin, Real-Time Systems, McGrow-Hill, 1997.

[2] C.H. Koo and H.H. Lee, “Distributed Simulator design by using of SimNetwork to overcome speed limit on GenSim”, Recent Advances in Space Technologies (RAST), 2011 5th International Conference on. pp. 430-435, 2011.

[3] S.H. Lee, A.R. Jo, H.J. Kim, H.M. Jo Y.S. Park and C.H. Lee, “Real-Time Communication for Military Test Equipments on Windows Systems” The Journal of the Korean Institute of Next Generation Computing, Vol 8, No.4, pp. 47-57, 2012.

[4] M.G. Ju, J.W. Lee, C.S. Jang, S.H. Kim and C.H. Lee, “A Method to Support Real-time for User-level Robot components on Windows”, The Journal of Korea Contents Association, Vol11, No7, 2011.

[5] M.G. Ju, S.H. Lee and C.H. Lee, “RTiK: Real-Time implant Kernel on Microsoft Windows”, TENCON 2011 – 2011 IEEE Region 10 Conf. Nov. 2011.

[6] C.I. Song, S.H. Lee, M.G. Ju and C.H. Lee, “Real-time processing function support on the multiprocessor Windows”, The Journal of the Korea Contents Association, Vol 12, No. 6, pp. 66-77, Jun. 2012.

[7] M.G. Ju, J.O. Lee, J.J. Kim, H.M. Jo, Y.S. Park and C.H. Lee, “A Method for real-time processing function support on the x86-based Windows”, The Journal of Korea Nextgeneration Computing, Vol. 11, No. 4, pp. 47-58. Aug. 2011.

[8] IntervlalZero, RTOS Platform, http://www.intervalzero.com (Access Date: 1 September, 2020).

[9] TENASYS, Embedded Virtualization Solutions, http://www.tenasys.com (Access Date: 14 September, 2020).

[10] J.W. Lee, M.H. Cho, J.J. Kim, H.M. Jo, Y.S. Park, C.H. Lee, “Design and Implementation of Real-time Implanted Kernel, RTiK to Support Real-Time for a Test Set based on

Windows", Journal of The Korea Contents Association, Vol. 10, No. 10, pp. 36-44, Oct. 2010.

[11] Intel, "Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture", September 2009.

[12] Intel, "Intel 64 Architectures x2APIC Specification", Intel, Intel 2008.

[13] David A. Solomon, Mark E. Russinovich, "Inside Windows 2000, Third Edition", Microsoft, 2000.

[14] Mark E. Russinovich and David A. Solomon, Microsoft Windows Internals 6th Edition, 2012.

[15] ILC DDC, MIL-STD-1553 Designer's Guide, 5th Ed., ILC Data Device Corporation, 1995.

[16] D.A. Godse and A.P. Godse, Microprocessors, Technical Publications Pune, 2007.

[17] O. Bailey, Embedded systems: desktop integration, Wordware Publishing, 2005.

[18] Intel, MultiProcessor Specification Version 1.4, Intel, 1997.

[19] Wikipedia, "Front-side-bus", http://en.wikipedia.org/wiki/Front-side-bus (Access date: 13 September, 2020).

[20] Microsoft, "Scheduling Priorities", http://msdn.microsoft.com/en-us/library/ms685100(VS.85).aspx (Access Date : 27 August, 2020).

8. Acknowledgements

I would like to thank my dissertation advisor Professor Lee Cheol-Hoon for his supervise and expert advice throughout long time, Lee Sang-Gil for his support and Professor Yeun Chan-Yeob for his advice, as well as CEO Muammar Al Fulaiti for extraordinary support and Dr. Khalid Al Kaabi for his encouragement.

