The Safety Feature Test of QNX RTOS

Jang Yeol Kim[a*] , Young Jun Lee[a]
Instrumentation and Control / Human Factors Division, Korea Atomic Energy Research Institute,
1045 Daedeok-daero, Yuseong-gu, Daejeon, Korea 305-353
**Corresponding author: jykim@kaeri.re.kr*

1. Introduction

Benchmarking is a point of reference by which something can be measured. The QNX is a kind of Real Time Operating System(RTOS) developed by QSSL(QNX Software Systems Ltd.) in Canada. The ELMSYS is the brand name of commercially available PC to be applied such as Cabinet Operator Module(COM) of Digital Plant Protection System(DPPS) and COM of Digital Engineered Safety Features Actuation System(DESFAS-AC). The ELMSYS PC Hardware will be qualified by KTL(Korea Testing Lab.) in order to use as a Cabinet Operator Module(COM). QNX RTOS is dedicating by KAERI now. This paper describes the outline and some safety features among benchmarking test for QNX RTOS under the ELMSYS PC platform.

2. Benchmark Methods and Results

The nuclear safety applications such as Cabinet Operator Module(COM) of Digital Plant Protection System(DPPS) and COM of Digital Engineered Safety Features Actuation System(DESFAS-AC) are constructed of commercial grade components procured on the basis of manufactures published data and guidance.

In order to use these systems, COTS dedication are required. But, most of commercial grade items are difficult to gather information. Therefore, we selected the special testing method for functional and performance requirement as a dedication methodology.

The methods applied for special testing of COTS software are compliant to the NRC Standard Review Plan (SRP) Appendix 7.0-A, Section C.3.8, Review of the acceptance of Commercial-Grade Digital Equipment and uses NUREG/CD-6421 as a guide. The SRP endorses EPRI TR-106439 noting that of the four methods described in EPRI NP-5652; methods 1,2,3 and 4 are the most likely required to qualify a commercial product for a protection system. This paper focuses on Method 1 – Special Testing and Inspection. : Benchmarking testing performed as Special Testing.

The QNX microkernel provides such core facilities as message passing, thread scheduling, timers, synchronization objects, and signals whereas the processes builds on the kernel facilities to provide additional process-level semantics, memory management, and pathname management. Optional extended service available include the file system, TCP/IP network protocols, and message queues. According to special test plan, test items are selected. Selected test items are executing as a special test under hardware and software testing environment.

2.1 Testing Environments

To perform the safety features test among benchmark test of QNX RTOS, it is very important to clearly established testing hardware and software environment. The hardware specification and software specification for testing are as shown in Table 1 and Table 2 respectively.

Table 1. Hardware Specification for testing

NO	Equipment	Model	Function	QTY	Precondition
1	Target PC	Processor	1 x Intel Pentium 4 (x86)	1	
		Reference Board	Desktop PC Motherboard	1	
		Clock speed	2.4 GHz	1	
		Memory	4GB	1	
2	Host PC	SAMSUNG Sens Notebook	(Microsoft Windows 2000 Professional, CPU 2.0 GB, Memory 1GB, HDD 70G)	1	Connected by Ethernet
3	Communication	3COM	Internal Communication between Target and Host	1	Connected by Hub
4	Checker	OS Checker	OS Checker Board	1	Interrupt Occurance Input and Output
5	PCI	PCI	PCI Card(DB-37 Femail)		
6	Interrupt		Interrupt Generator(MCU: Atemega 128) RS232, DB-9 Femail		

Table 2. Software Specification for testing

NO	SW Name	Functions
1	Neutrino V6.4	Execution environment of Real Time Operating System
2	Momentics V4.4	QNX Programming Development and Download of Test Program
3	Capture S/W	Capture of Test Result

2.2 Test Items

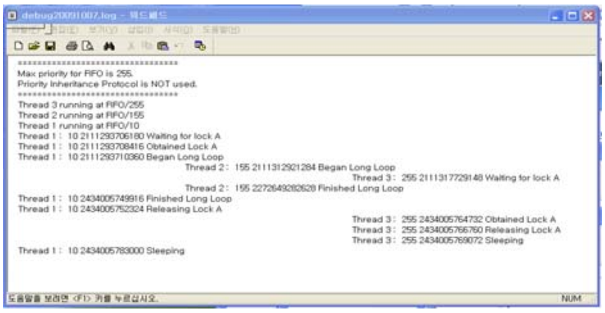
It was selected test items for functional requirement and performance requirement including safety features such as priority inversion/inheritance and deadlock avoidance like Table 3. Testing of functional and performance will continue the execution later.

Table 3. Test items for safety features

Critical characteristics	Test Items
1.Priority Inversion	1.1 Priority Inversion
	1.2 Priority Inheritance
2. Deadlock Avoidance	2.1 Deadlock
	2.2 Deadlock Avoidance

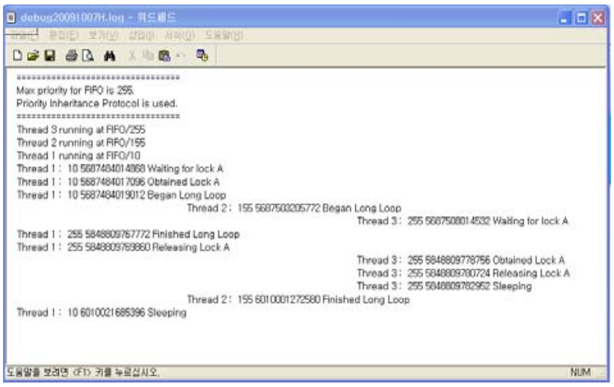
2.3 Test Results

It was the result of testing on priority inversion(Figure 1) and deadlock(Figure 3). The QNX RTOS has priority inheritance protocol(Figure 2). Through the test of priority inversion, QNX RTOS has been identified the facility of the priority inheritance mechanism. In case of deadlock, there is almost no solution. However, using a timer function with locking mechanism can be solved(Figure 3).


debug20091007.log - 워드패드
파일(F) 편집(E) 보기(V) 삽입(I) 서식(O) 도움말(H)
도움말을 보려면 <F1> 키를 누르십시오.
NUM

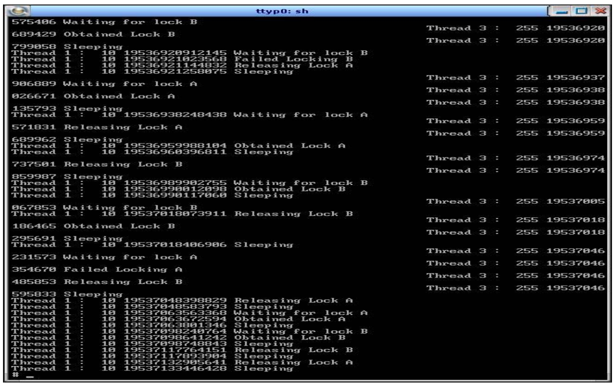
```
**********************************
Max priority for FIFO is 255.
Priority Inheritance Protocol is NOT used.
**********************************
Thread 3 running at FIFO/255
Thread 2 running at FIFO/155
Thread 1 running at FIFO/10
Thread 1 :  10 2111293706180 Waiting for lock A
Thread 1 :  10 2111293708416 Obtained Lock A
Thread 1 :  10 2111293710360 Began Long Loop
                              Thread 2 :  155 2111312921284 Began Long Loop
                                                              Thread 3 :  255 2111317729148 Waiting for lock A
                              Thread 2 :  155 2272649282628 Finished Long Loop
Thread 1 :  10 2434005749916 Finished Long Loop
Thread 1 :  10 2434005752324 Releasing Lock A
                                                              Thread 3 :  255 2434005764732 Obtained Lock A
                                                              Thread 3 :  255 2434005766760 Releasing Lock A
                                                              Thread 3 :  255 2434005769072 Sleeping
Thread 1 :  10 2434005783000 Sleeping
```

Figure 1. Not applied for Inheritance Mechanism


debug20091007H.log - 워드패드
파일(F) 편집(E) 보기(V) 삽입(I) 서식(O) 도움말(H)
도움말을 보려면 <F1> 키를 누르십시오.
NUM

```
**********************************
Max priority for FIFO is 255.
Priority Inheritance Protocol is used.
**********************************
Thread 3 running at FIFO/255
Thread 2 running at FIFO/155
Thread 1 running at FIFO/10
Thread 1 :  10 5687484014868 Waiting for lock A
Thread 1 :  10 5687484017096 Obtained Lock A
Thread 1 :  10 5687484019012 Began Long Loop
                              Thread 2 :  155 5687503205772 Began Long Loop
                                                              Thread 3 :  255 5687508014532 Waiting for lock A
Thread 1 :  255 5848809767772 Finished Long Loop
Thread 1 :  255 5848809769860 Releasing Lock A
                                                              Thread 3 :  255 5848809778756 Obtained Lock A
                                                              Thread 3 :  255 5848809780724 Releasing Lock A
                                                              Thread 3 :  255 5848809782952 Sleeping
                              Thread 2 :  155 6010001272580 Finished Long Loop
Thread 1 :  10 6010021685396 Sleeping
```

Figure 2. Applied for Inheritance Mechanism



```
575406 Waiting for lock B
                                                     Thread 3 :   255 19536920
689429 Obtained Lock B
                                                     Thread 3 :   255 19536920
799058 Sleeping
Thread 1 :   10 19536920912145 Waiting for lock B
Thread 1 :   10 19536921023568 Failed Locking B
Thread 1 :   10 19536921144832 Releasing Lock A
Thread 1 :   10 19536921258075 Sleeping
                                                     Thread 3 :   255 19536937
906889 Waiting for lock A
                                                     Thread 3 :   255 19536938
026671 Obtained Lock A
                                                     Thread 3 :   255 19536938
135793 Sleeping
Thread 1 :   10 19536938248438 Waiting for lock A
                                                     Thread 3 :   255 19536959
571831 Releasing Lock A
                                                     Thread 3 :   255 19536959
689962 Sleeping
Thread 1 :   10 19536959988104 Obtained Lock A
Thread 1 :   10 19536960396811 Sleeping
                                                     Thread 3 :   255 19536974
737501 Releasing Lock B
                                                     Thread 3 :   255 19536974
859987 Sleeping
Thread 1 :   10 19536989902755 Waiting for lock B
Thread 1 :   10 19536990012098 Obtained Lock B
Thread 1 :   10 19536990117060 Sleeping
                                                     Thread 3 :   255 19537005
067853 Waiting for lock B
Thread 1 :   10 19537018073911 Releasing Lock B
                                                     Thread 3 :   255 19537018
186465 Obtained Lock B
                                                     Thread 3 :   255 19537018
295691 Sleeping
Thread 1 :   10 19537018406906 Sleeping
                                                     Thread 3 :   255 19537046
231573 Waiting for lock A
                                                     Thread 3 :   255 19537046
354670 Failed Locking A
                                                     Thread 3 :   255 19537046
485853 Releasing Lock B
                                                     Thread 3 :   255 19537046
595833 Sleeping
Thread 1 :   10 19537048398829 Releasing Lock A
Thread 1 :   10 19537048583793 Sleeping
Thread 1 :   10 19537063563368 Waiting for lock A
Thread 1 :   10 19537063672594 Obtained Lock A
Thread 1 :   10 19537063801346 Sleeping
Thread 1 :   10 19537098240764 Waiting for lock B
Thread 1 :   10 19537098641242 Obtained Lock B
Thread 1 :   10 19537098748843 Sleeping
Thread 1 :   10 19537117764151 Releasing Lock B
Thread 1 :   10 19537117893904 Sleeping
Thread 1 :   10 19537132905641 Releasing Lock A
Thread 1 :   10 19537133446428 Sleeping
# _
```

Figure 3. Deadlock Occurrence and Recovery

3. Conclusions

Our benchmarking testing environments have been well set up between host and target. This paper described the test result of safety features such as deadlock and priority inversion on the QNX RTOS for a specific system configuration under ELMSYS PC.

The benchmarking testing as a special test for QNX RTOS will continue under the ELMSYS PC.

REFERENCES

[1] QNX Software Systems, QNX Neutrino Realtime OS: Kernel Benchmark Methodology.

[2] Standard Review Plan, NUREG-0800, Revision 4, Chapter 7, Instrumentation and Controls – Overview of Review Process

[3] NUREG/CR-6421, A Proposed Acceptance Process for Commercial Off-the-Shelf(COTS) Software in Reactor Applications

[4] Electric Power Research Institute(EPRI) NP-5652, Guideline for the Utilitization of Commercial Grade Items in Nuclear Safety Related Applications

[5] EPRI Topical Report, TR-106439, Guideline on Evaluation and Acceptance of Commercial Grade Digital Equipment for Nuclear Safety Related Applications

[6] IEEE Std 7-4.3.2-1993, IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations.