HITACHI
Inspire the Next

# Evaluation of uClinux and PREEMPT_RT for Machine Control System

2014/05/20

Hitachi, Ltd. Yokohama Research Lab
Linux Technology Center

**Yoshihiro Hayashi**
yoshihiro.hayashi.cd@hitachi.com

Yokohama Research Lab.
Linux Technology Center

# Agenda

1. Background
2. Experience from appling
   both PREEMPT_RT and uClinux
3. Evaluation of PREEMPT_RT and uClinux
   and Environment
4. Conclusion

HITACHI
Inspire the Next

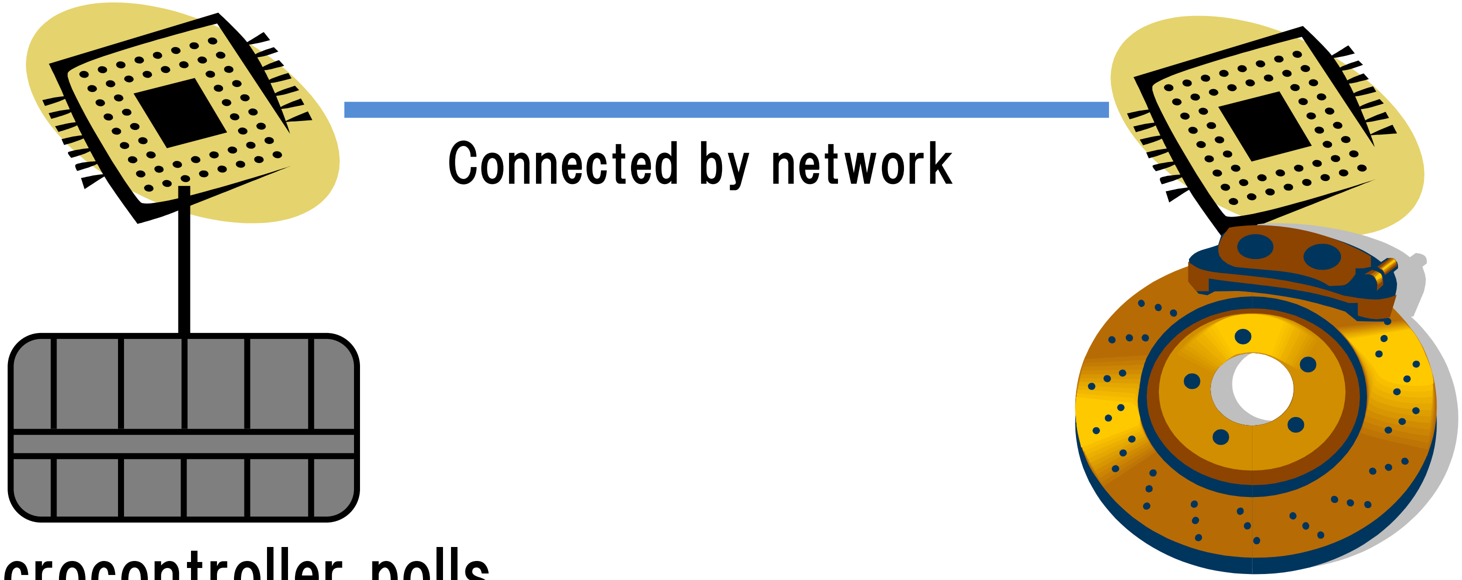
# 1. Background

Yokohama Research Lab.
Linux Technology Center

# What is a Machine Control System?

A Realtime System reads and writes data at microsecond to millisecond intervals, to control physical machines

## Example: A brake system in a car



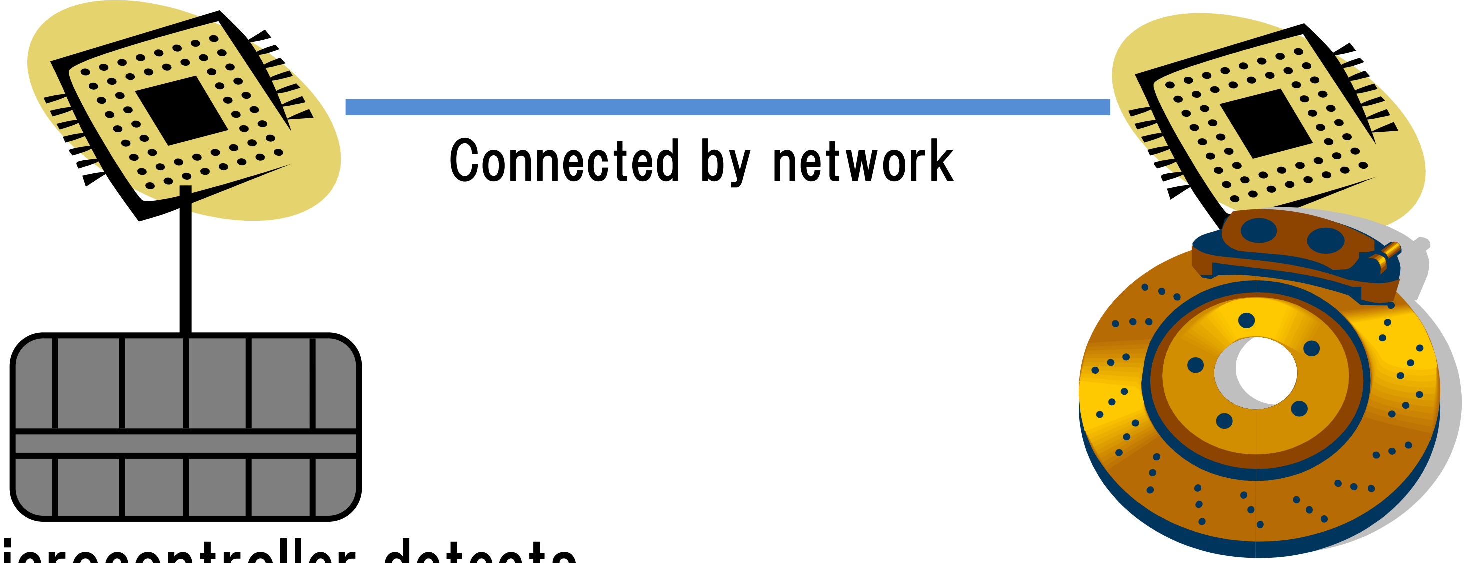
Connected by network

A microcontroller polls
the sensor value to detect
how strong the pedal
is pushed and sends the
information over the network

Another microcontroller
receives the information
and applies the brake

# Problem without Realtimeness

## Example: A brake system in a car



If the microcontroller detects a push of pedal with some delay because of other programs, sending the information is also delayed.

Detection of the sensor value and applying brake is also delayed.

It takes longer to apply brake after the brake pedal is pushed.

Braking Distance Gets Longer

# Issues of Machine Control System Development

**Issues: Too large variety of environments for development**
**Linux can be used as the single unified environment**

## Current Machine Control Systems Development

- Use of different OSes for different products, or even no OS
  - VxWorks, FreeRTOS, μITRON, μT-Kernel etc.
- Issues
  - Small number of application developers for each OS
  - Difficult to acquire solid know-how
  - No middleware or network stack (or proprietary if available)

With Linux

The Issues are solved

- Lots of application developers for Linux
- Linux know-how is easier to acquire
- Multiple middleware and network stacks are included in Linux

Applying Linux to Machine Control Systems
still has other issues

# Issues of applying Linux to Embedded Systems

Lack of realtimeness, and disk and memory usage are the issues.

Linux is good at batch loads that process large amount of data.
Linux is originally for PCs. PCs have faster CPUs and larger memory and disks, compared to Machine Control Systems.

| | | |
|---|---|---|
| Issues | Lack of realtimeness | Memory and disk usage |
| Solutions | A mechanism to respond faster to the external inputs (PREEMPT_RT)<br>A mechanism to place code and data used in IRQ in SRAM | Linux for small controllers (uClinux) |

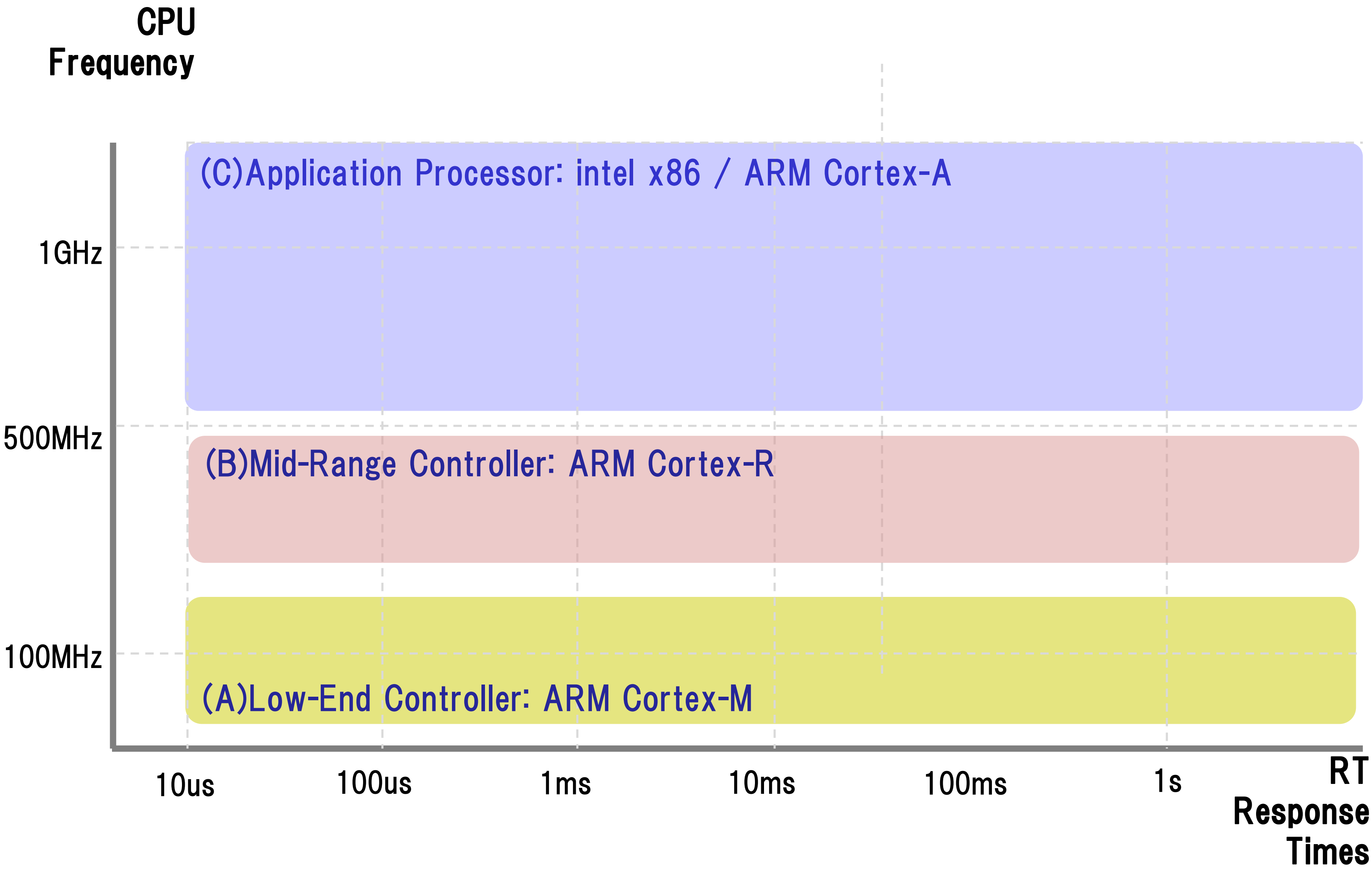
# PREEMPT_RT

- Huge patchset
  - Making in-kernel locking-primitives (using spinlocks) preemptible through reimplementation with rtmutexes.
  - Implementing priority inheritance for in-kernel spinlocks and semaphores.
  - Converting interrupt handlers into preemptible kernel threads.

# uClinux

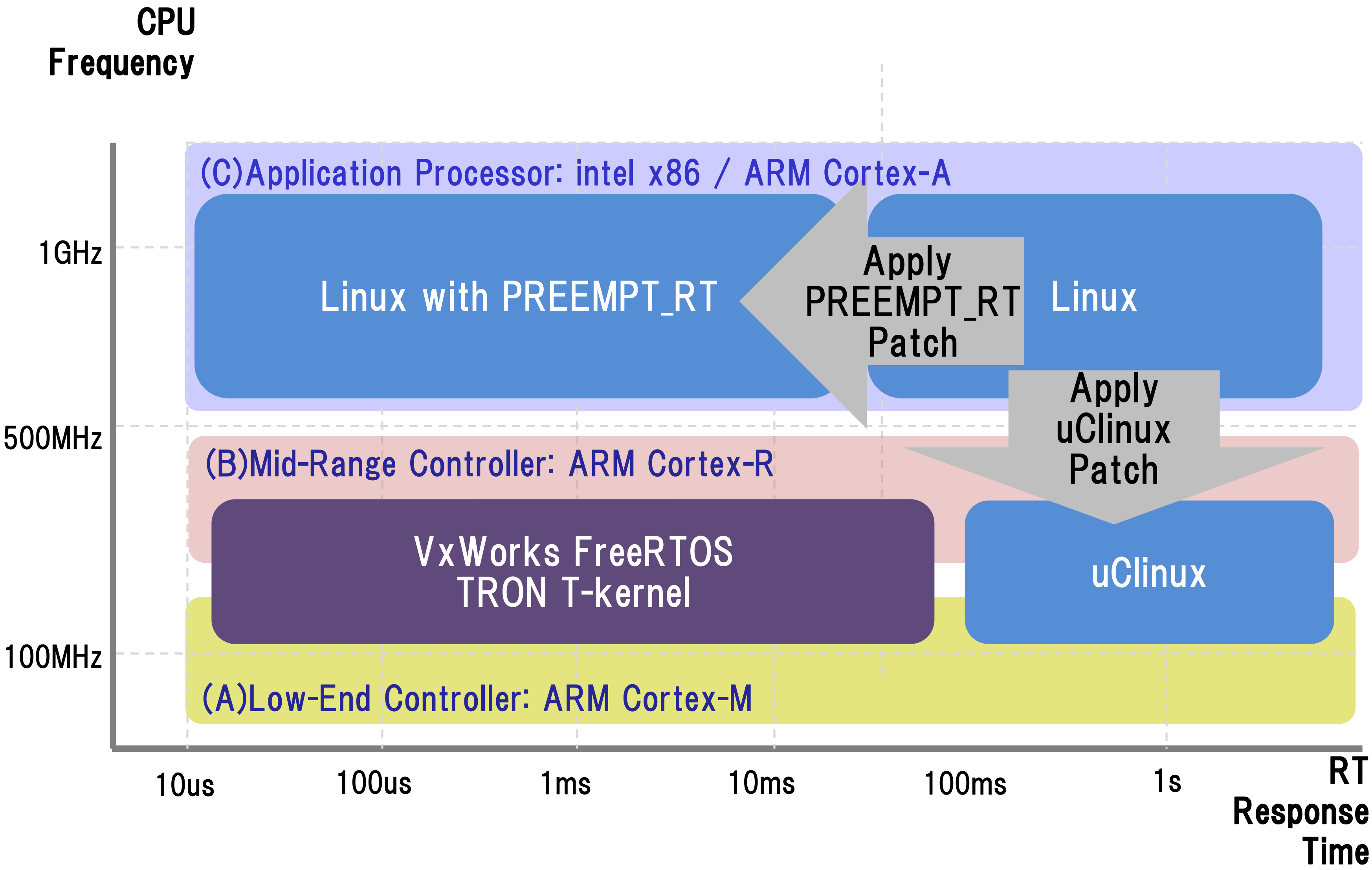
- Huge patchset / distoribution
  - – “Mu” stands for “micro”, and “C” is for “controller”.
  - – Name of the patchset suitable for NO MMU micro controllers.
  - – Name of the distribution which includes a userland library and basic commands.

# CPU Category



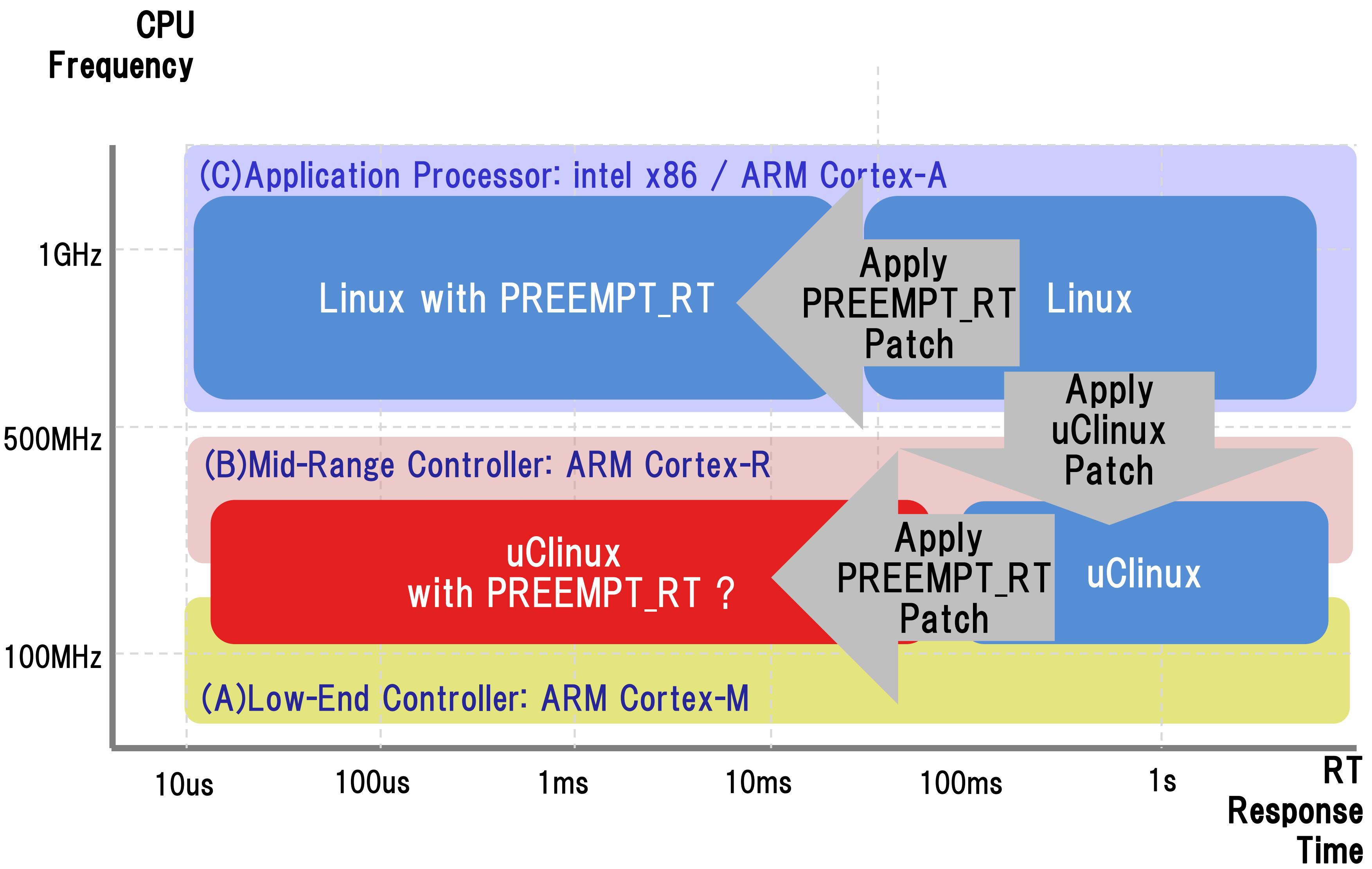
Yokohama Research Lab.
Linux Technology Center

# CPU Category and Coverage of Linux



Yokohama Research Lab.
Linux Technology Center

# My Challenge



Yokohama Research Lab.
Linux Technology Center

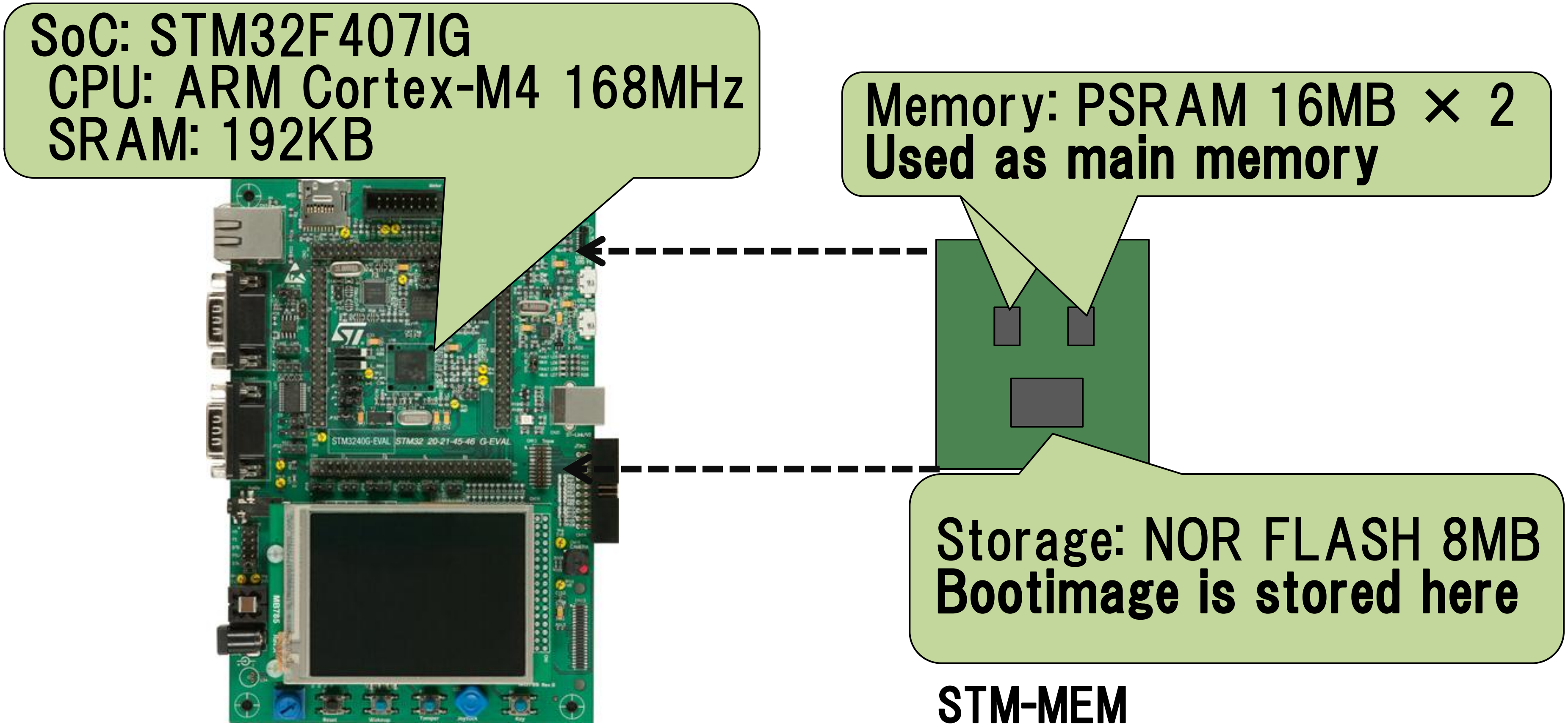
# 2. Experience from applying both PREEMPT_RT and uClinux

0. My evaluation environment

1. Lack of drivers for peripherals
2. Difficulty of applying multiple huge patchsets
3. Unsuitability of in-kernel debugging tools

# My evaluation environment

- Linux STM3240G-EVAL Kit made by EMCRAFT



SoC: STM32F407IG
CPU: ARM Cortex-M4 168MHz
SRAM: 192KB

Memory: PSRAM 16MB × 2
Used as main memory

Storage: NOR FLASH 8MB
Bootimage is stored here

STMicroelectronics
STM3240G-EVAL Board

STM-MEM
plug-in board

- **uClinux and BSP(Board Support Package) are available based on 2.6.33**

# Lack of drivers for peripherals

- In general…
  - BSPs(board support packages) are developed out of tree; have just a small amount of users; are not mature.
  - Users should evaluate if its functions and quality are satisfactory before working on it.
- In my evaluation…
  - hrtimer is necessary for realtime application but not available on evaluation environment
    - “STM32 System Timer” in SoC is used as clockevent. However, “oneshot mode” was not implemented.
    - “oneshot mode” is necessary for hrtimer.
    - I implemented “oneshot mode” and made hrtimer available.

# Difficulty of applying multiple huge patchset

- In general…
  - Each patchset can be applied onto a certain version of vanilla kernel.
  - Patchsets are not designed to be used with other patchsets.
- In my evalutation…
  - I applied PREEMPT_RT patch on BSP which included uClinux then many hunks were rejected.
  - I applied rejected hunks manually.
  - I hope that  at least one patchset will be upstreamed.
    - and the other one will be compatible with the upstreamed one.

Yokohama Research Lab.
Linux Technology Center

# Unsuitability of in-kernel debugging tools

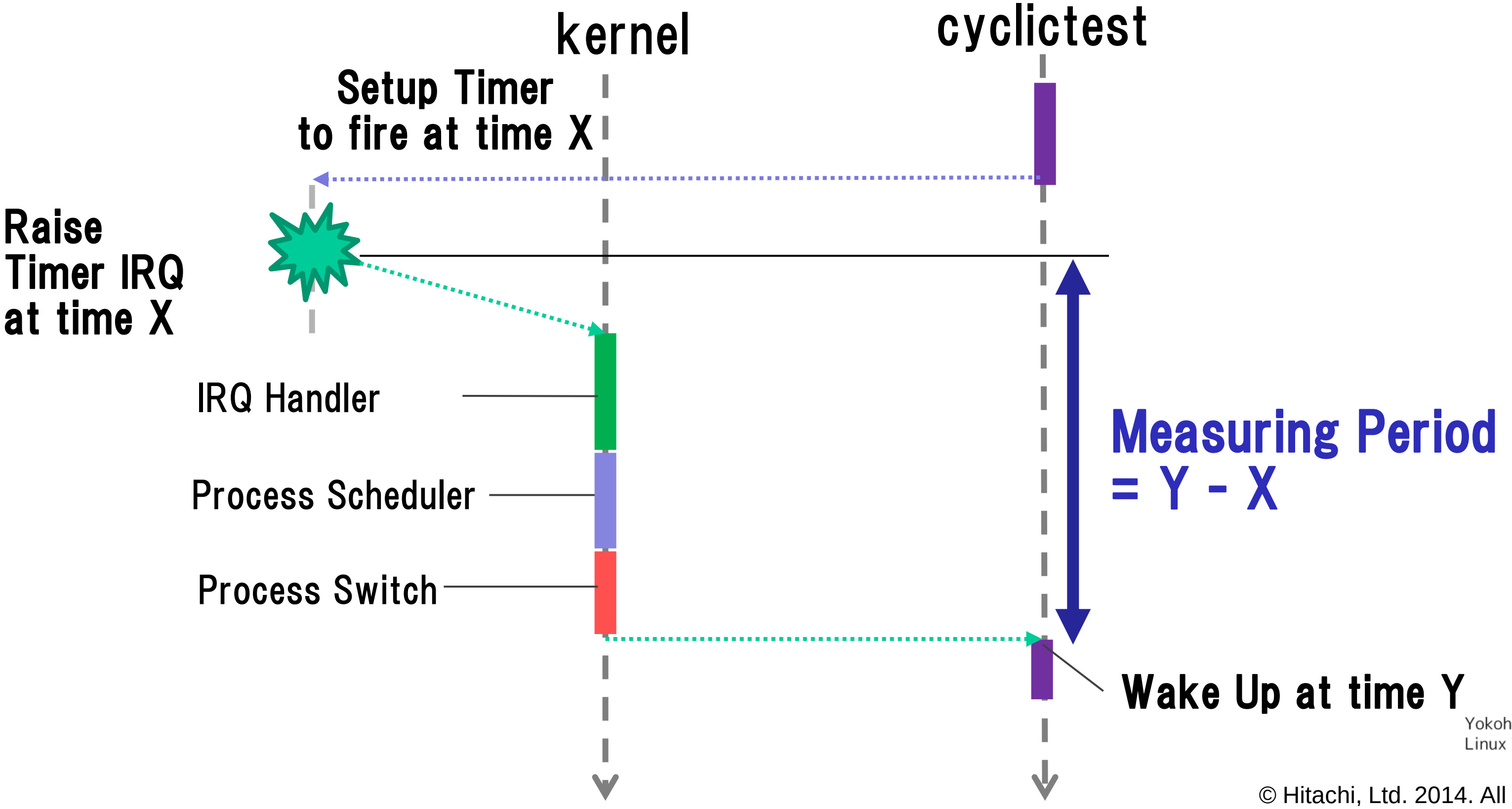
- In general…
  - – Many in-kernel debug tools are unsuitable for uClinux running in Cortex-M
    - In-kernel debug tools are too heavy for Cortex-M
    - Some of the tools depend on architecture specific functions
- In my evaluation…
  - – ftrace was too heavy.
  - – IRQ tracer did not seem to work.
  - – I had to “reinvent” necessary debugging tools.
    - I implemented a simple tracer which stores traces in SRAM.

# Evaluation of PREEMPT_RT, uClinux and environment

- Evaluation1: Cyclictest
- Evaluation2: Memory Access

# Cyclictest

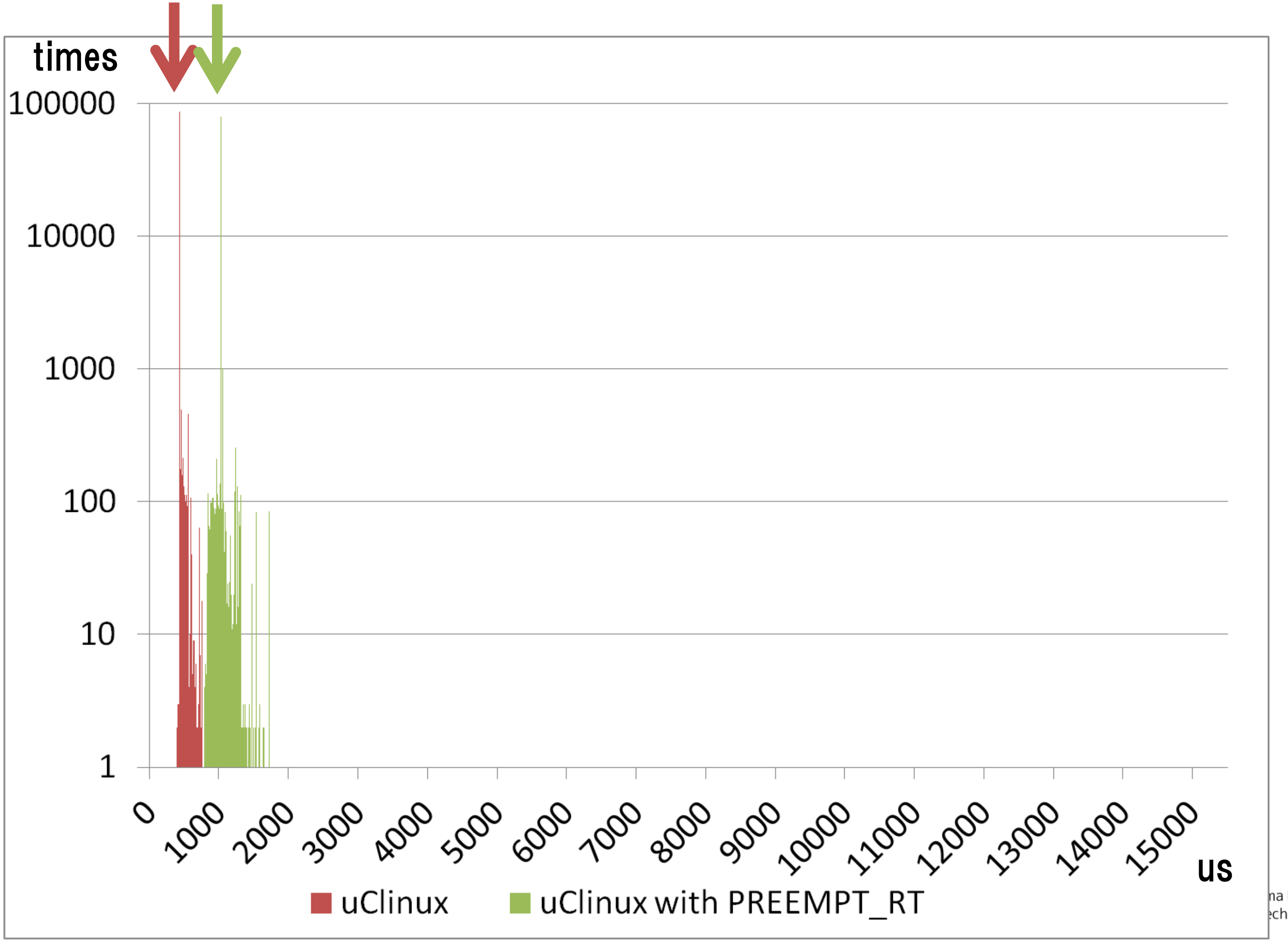
- For realtime applications, latency of timer IRQ response is very important
  - Realtime applications use timer to implement periodical procedures.



# Result of Cyclictest (1/3)

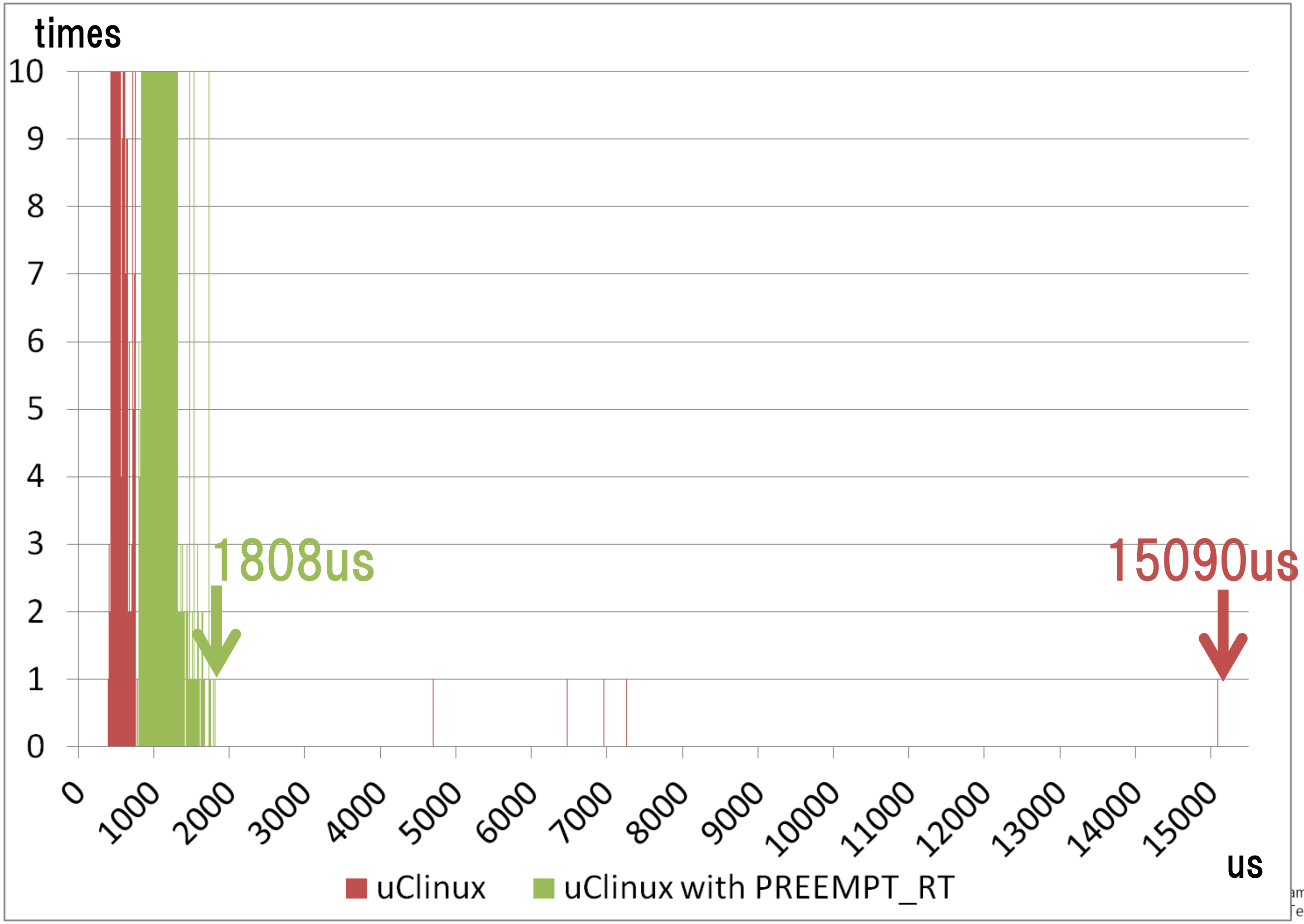
## Average

440us 1024us



# Result of Cyclictest (2/3)

## Max Latency



# Result of Cyclictest (3/3)

- Max latency is very important for realtime systems because developers have to define deadlines for periodical procedures.
- uClinux is better in worst case with PREEMPT_RT than without it.

| | uClinux | uClinux with PREEMPT_RT |
|---|---|---|
| Average | 440us | 1024us |
| Worst case | 15090us | **1808us** |

# Memory Access PSRAM vs SRAM

- In my evaluation environment, uClinux run in PSRAM on the plug-in board.
- The SoC has just 192kb SRAM. This is too small to run uClinux.

- PSRAM on plug-in board is far slower than SRAM in SoC.
- If code and data used in IRQ context are placed in SRAM, IRQ latency will be lower.

# Result of Memory Access PSRAM vs SRAM

- Evaluated how much faster is SRAM than PSRAM in my environment.
  - Used own micro benchmark, which just dose 10M integer additions in-memory.

| | Code is stored in | Data is stored in | Time (Shorter is better) |
|---|---|---|---|
| Case 1 | PSRAM | PSRAM | 12047511 us |
| Case 2 | PSRAM | SRAM | 10430945 us |
| Case 3 | SRAM | PSRAM | 2396412 us |
| Case 4 | SRAM | SRAM | 651154 us |

about 20 times faster

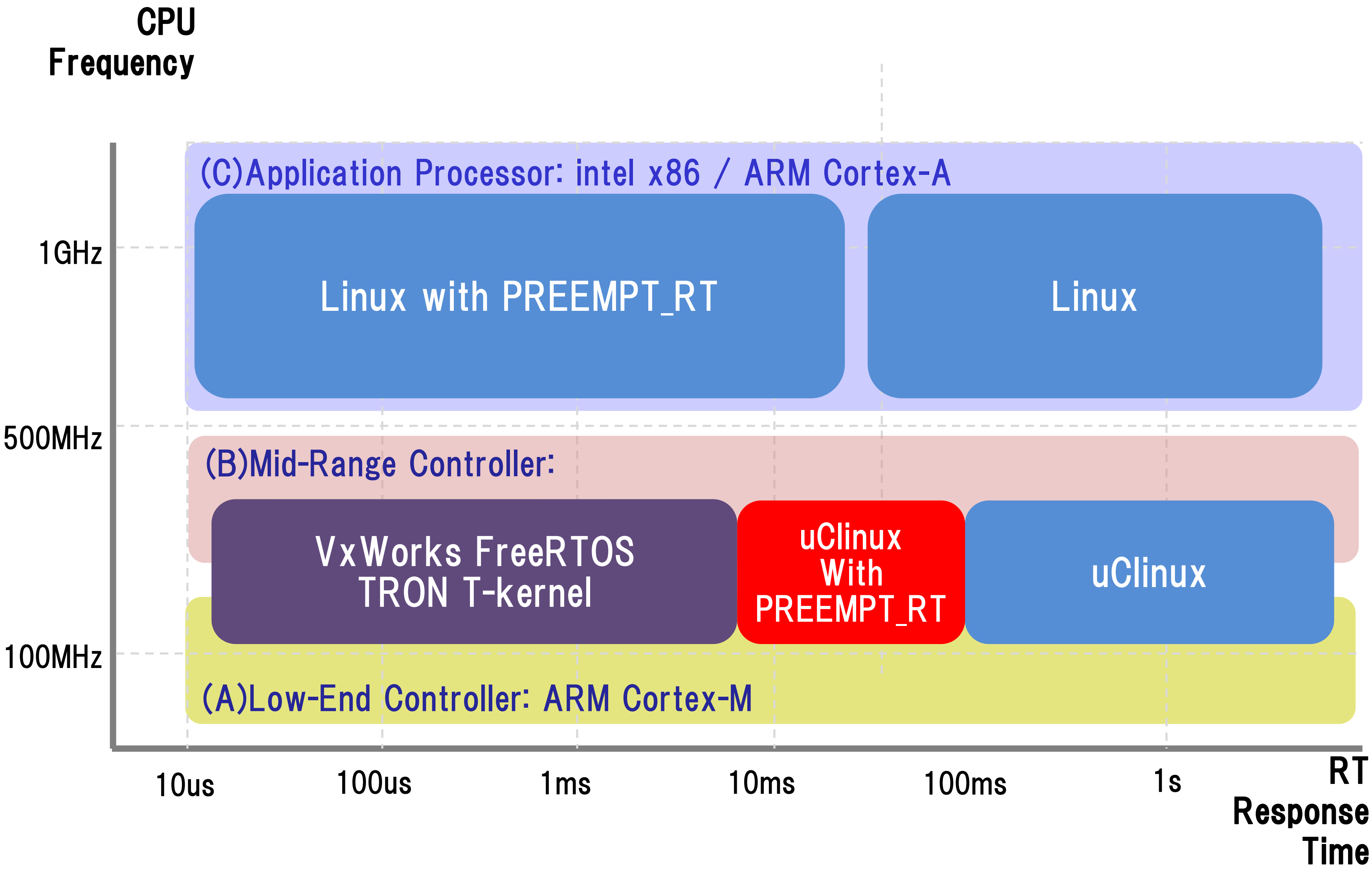
  - If all the code and data of the IRQ context are placed in SRAM, timer IRQ latency will be probably 20 times lower.
  - Then max latency will be 90us instead of 1808us.

HITACHI
Inspire the Next

# Conclusion

Yokohama Research Lab.
Linux Technology Center

# Coverage of Linux after my Challenge



Yokohama Research Lab.
Linux Technology Center

# Conclusion

- uClinux’s IRQ latency is shorter with PREEMPT_RT
  - Cyclictest showed that max latency was 1808us
  - Combination of uClinux and PREEMP_RT is useful for developing Machine Control Systems.
  - Max latency could be 90us if we would use SRAM for code and data of IRQ context.
- It’s very tough to apply both uClinux and PREEMPT_RT.
  - In-kernel debugging tools are too heavy for Cortex-M.
  - Upstreaming is important for ease of use.

# appendix

# Microbenchmark

```
int readfunc(void *addr, int length, int loop){
        int ret = 0;
        void *i;
        void *end;

        end = addr + length;
        while(loop-- > 0){
                for(i=addr; i<end; i++){
                        ret += *(char*)i;
                }
        }
        return ret;
}
```