BG003 : Renesas Technical Training

Development Tools for Micro Computer Products

Thursday, October 5, 2017
Thang Nguyen
Software Tool Solution 1 Group,
Software Solution 1 Section
Software Engineering Department
Renesas Design Vietnam Co., Ltd.

Contents

- □ Words in slides
- □ Embedded system overview
- □ Embedded system development flow
- □ Operating System
- □ Real-time Operating System
- □ Integrate development environment
- □ Cross software
- □ Simulation tools
- □ Evaluation boards
- □ Emulation tools

Words in slides

Words in slides 1

❑ User

- o Renesas sells semiconductor devices to our customers, such as electronics products manufacturer, automotive manufacturer, industrial products manufacturer, etc.
- o Customers design, manufacture, and sell their products to end customers.
- o Sometimes we take the word ‘user’ as customer, not end customer.
  - ▪ User system (target system): Our customer’s system. Electronic system.
  - ▪ User program (target program): Our customer’s program.
  - ▪ User interface: Interface between customer’s system and our (Renesas) tool

❑ MCU and MPU

- o MCU : Micro Controller Unit. Integrated ‘single chip’ device, such as M16C, H8/H8S, SH-2(2A) CPU, memory(ROM and RAM), and IO in one silicon
- o MPU : Micro Processing Unit Integrated device including CPU, cache, and IO, such as SH-4(4A). External large size memory, Flash and DRAM

Words in slides 2

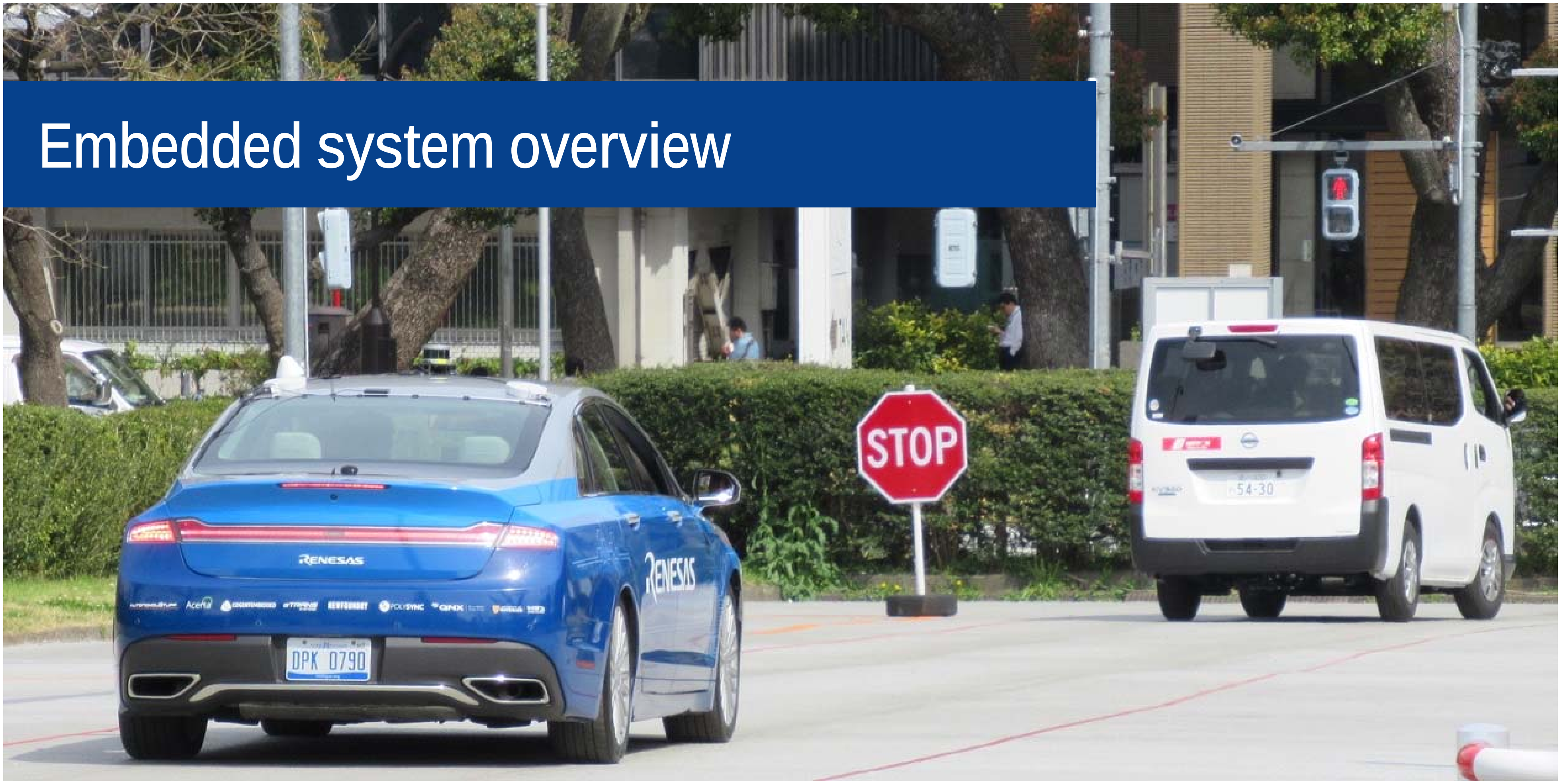
- ❑ Firmware
  - o Original meaning is ‘software stored in some hardware’ which is hard to rewrite, ie. software stored in ROM. In old days, memory is clearly divided into two type, one is RAM and the other one is ROM.
  - o Recently flash memory is used widely for software storage, and due to above historical reason, software stored in flash memory is also called ‘firmware’ in some case.
- ❑ Middleware
  - o Original meaning is ‘some software which is placed between higher layer and lower layer’.
  - o In embedded application, higher layer means ‘application’, and lower layer means ‘driver’.
  - o Example of middleware: audio/video codec (WMA / WMV / MP3 / MPEG / H.264), telecommunication codec (G.72x), protocol stack (TCP/IP), file system.

Words in slides 3

- ❑ BSP (Board Support Package)
  - o In embedded system, wide variety of different hardware (IO) is used, there is need for software package which is dedicated to specific hardware platform. Software package for such dedicated hardware is called ‘BSP’, as this is prepared for specific board.
- ❑ IPL (Initial Program Loader)
  - o IPL is firmware software which runs just after hardware power on.
- ❑ BIOS (Basic Input/Output System)
  - o BIOS is firmware software for IBM-PC compatible computer which runs just after hardware power on. This program identify and initiate IO hardware, such as HDD, CD drive, etc., and provide standard IO access software interface to higher level software, such as Windows OS.
  - o BIOS is very close to compact/simple OS(Operating System), but BIOS is not OS.
- ❑ CMOS (Complementary Metal-Oxide Semiconductor)
  - o P-MOS and N-MOS pair semiconductor digital logic circuit. Commonly used for current digital semiconductor devices. This word has no relation to any Operating System.

Words in slides 4

- ❑ MMU (Memory Management Unit)
  - o A part of CPU. Provides address translation from logical (virtual) address to physical address
- ❑ OS (Operating System)
- ❑ RTOS (Real-Time Operating System)
  - o Operating System for embedded application.
  - o Hard real-time, multi-task
- ❑ Real-time
  - o One of major demand in embedded system.
  - o Time sensitive. Something must be done within pre-determined specific time period.
- ❑ Task: a part of application. Usually task runs on the device without MMU.
- ❑ Process: a part of application. Usually process runs on the device with MMU



Embedded system overview

Words in slides 4

- ❑ An embedded system is a **computer system with a dedicated function** within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as **part of a complete device** often including hardware and mechanical parts

(Source: wikipedia.org)

Words in slides 4

❑ An embedded system is a **computer system with a dedicated function** within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as **part of a complete device** often including hardware and mechanical parts

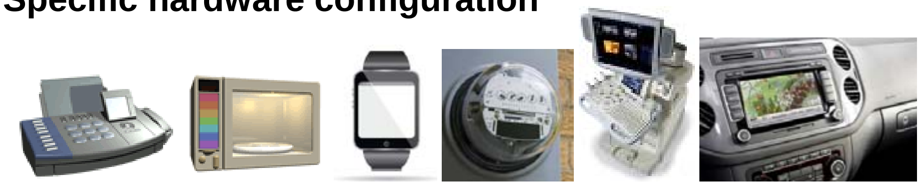
(Source: wikipedia.org)

❑ Embedded systems are **very popular** nowadays

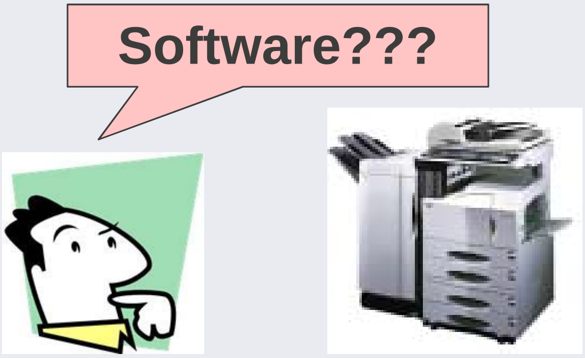
Differences between PC and Embedded System

PC

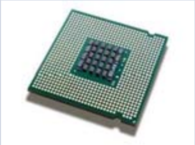
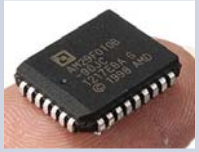
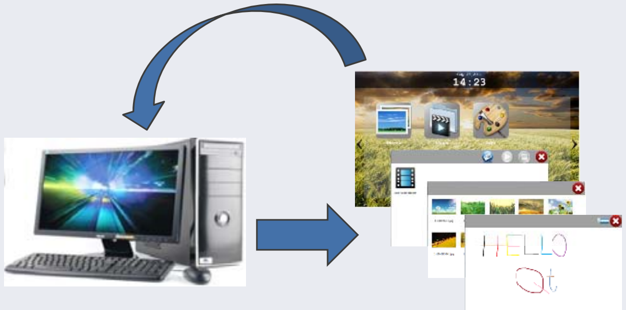
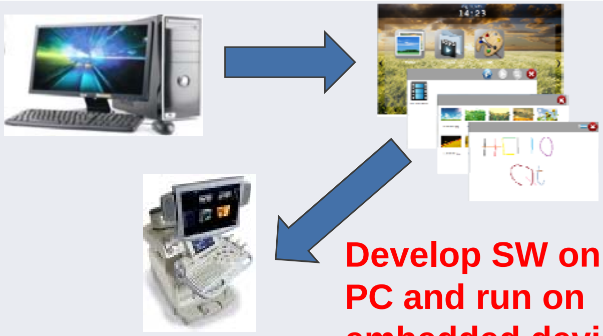
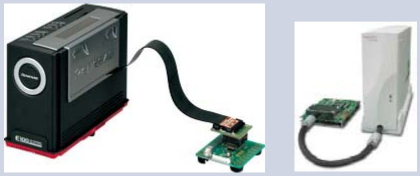
Embedded System

General usage	Dedicated function usage
Standard hardware configuration 	**Specific hardware configuration** 
- Easy to install OS or new Software 	**- Cannot change OS.** **- Need support of the Manufacturer to install new OS** - Difficult to update Software.

Differences between PC and Embedded System

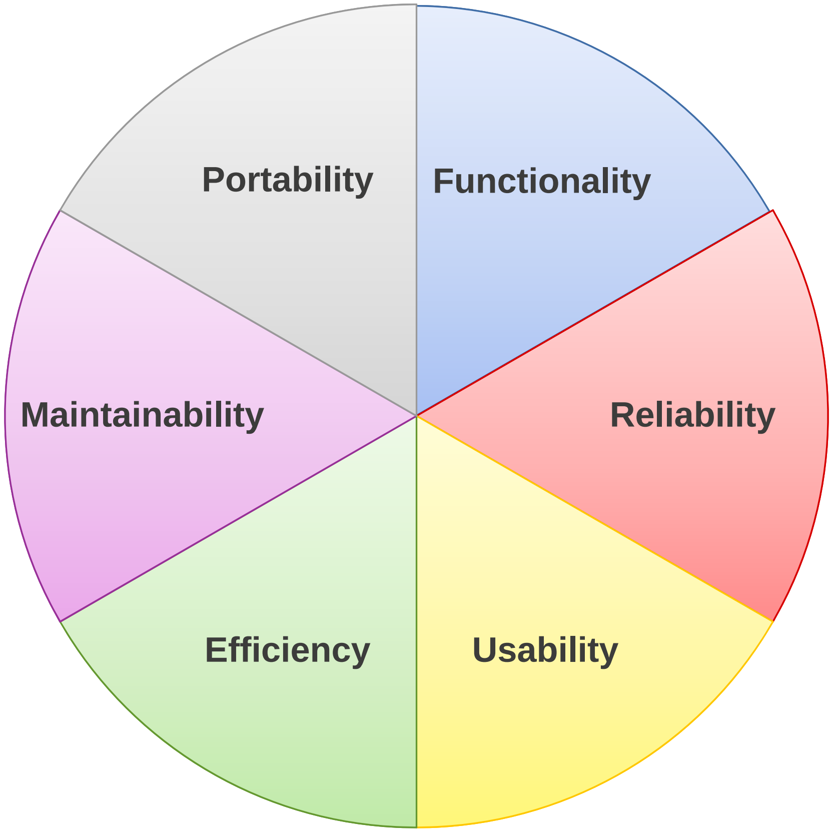
	Computer Software Dev.	Embedded Software Dev.
Value of performance	**Total data throughput** • CPU, GPU ??? • RAM ??? • HDD, SSD ??? • Gigaflop/Teraflop ???	**Real-time performance** • Time constraint ??? • Microsecond/Nanosecond ???
Software Visibility	**Software are visible** 	**Almost users do not know which software is running** 

Differences between PC and Embedded System

	Computer Software Dev.	Embedded Software Dev.
Hardware resources to run SW	**Rich hardware resources**  GHZ GB TB	**Limited hardware resources** MHZ KB/MB  MB
Development environment	 Develop and run Software on PC	 Develop SW on PC and run on embedded device
Debugging support	**No need special hardware** Can use PC to debug Software	Need special **Hardware** 

Differences between PC and Embedded System

- How much QUALITY is important in Embedded System?



Reference Numbers (Typical Example) - Architecture Comparison

Architecture	8 Bit CPU	16 Bit CPU	32 Bit CPU
ALU data width	8 bit	16 bit	32 bit
Maximum Linear Address Space	64K (16 bit addressing)	16M (24 bit addressing)	4G (32 bit addressing)
Implemented Address Space (typical)	up to 64 K (on silicon)	128K to 1M or more (on silicon)	64M to 512M (external)
MMU (OS)	No	No	Yes
CPU Clock	10MHz	40 MHz	400MHz

Typical OS kernel size:

RTOS (uITRON)	Linux
10 ~ 20 KB	1 ~ 4 MB

ALU: Arithmetic logic unit
MMU: Memory management unit

Real-time operating systems (RTOS)

- ❑ Real-time operating system (RTOS) is an operating system (OS) intended to serve real-time applications which process data as it comes in, typically without buffering delays.
- ❑ Key factors in a real-time OS are minimal interrupt latency and minimal thread switching latency; a real-time OS is valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time.
- ❑ Common RTOS:

QNX | BlackBerry Subsidiary

VxWorks

Itron

freeRTOS

Microsoft Windows CE

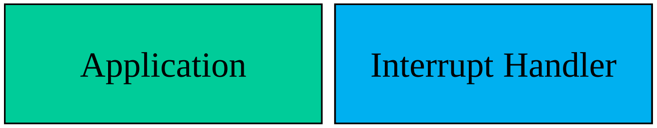
µC/OS-III™ The Real-Time Kernel

Software Structure (Typical Example)

1. Just One component

All Software

2. Application and Interrupt



Application

Interrupt Handler

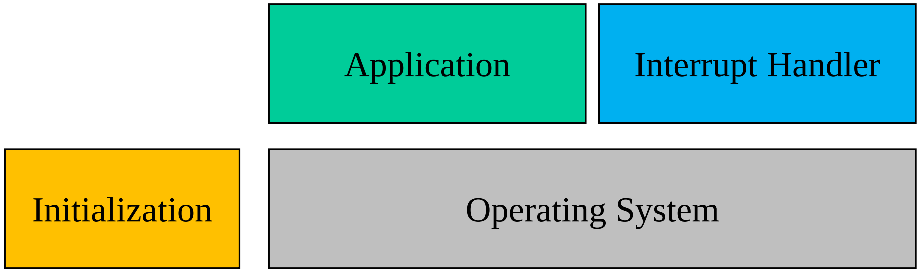
3. Initialization, Application and Interrupt

Initialization

Application

Interrupt Handler

4. Initialization, Operating System, Application and Interrupt



Application

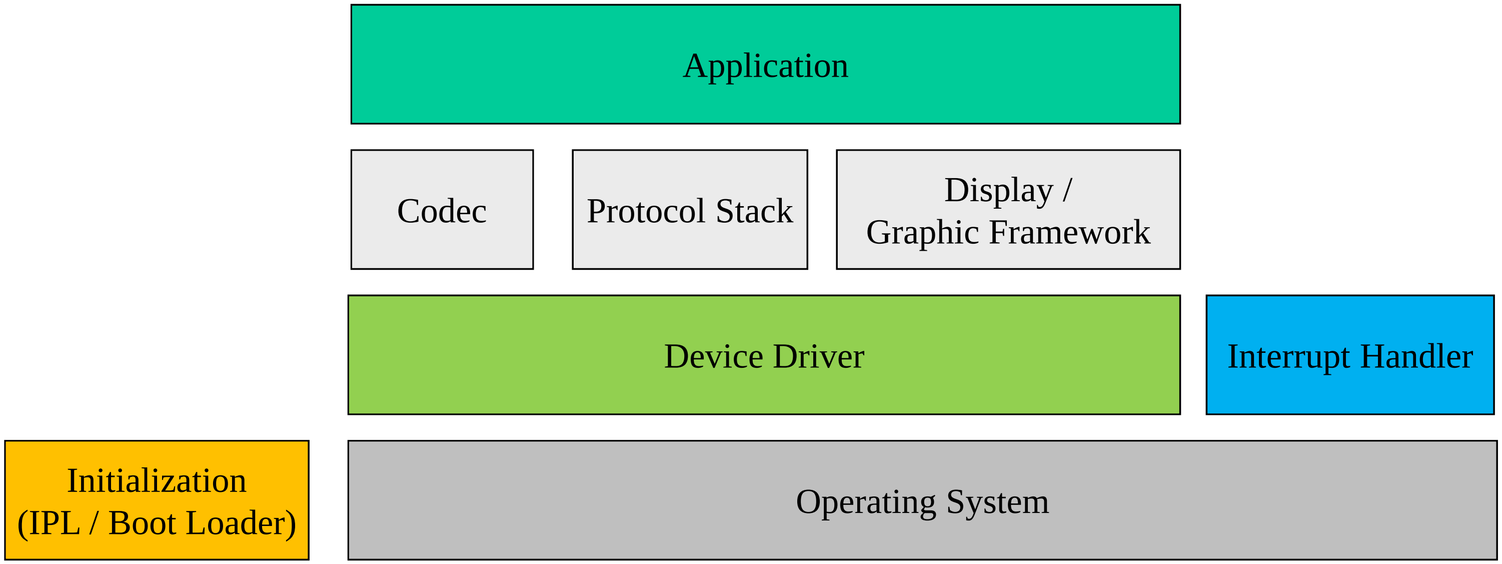
Interrupt Handler

Initialization

Operating System

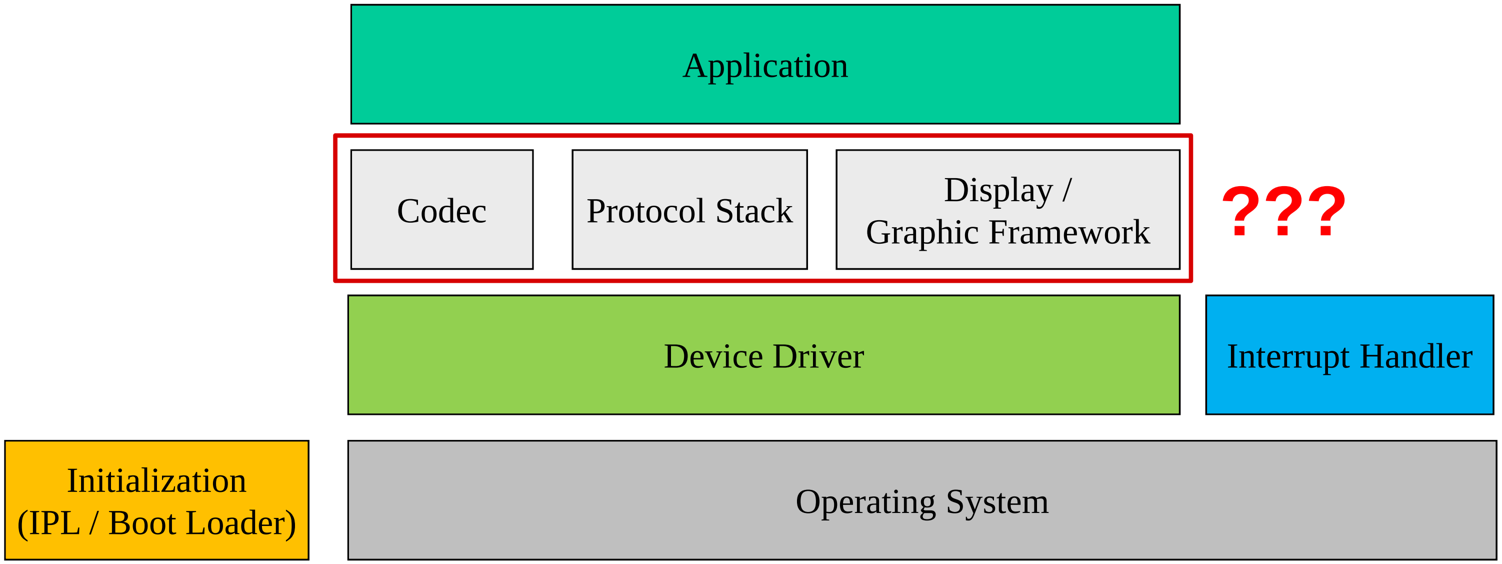
Software Structure (Typical Example)

5. More Complex Software Structure

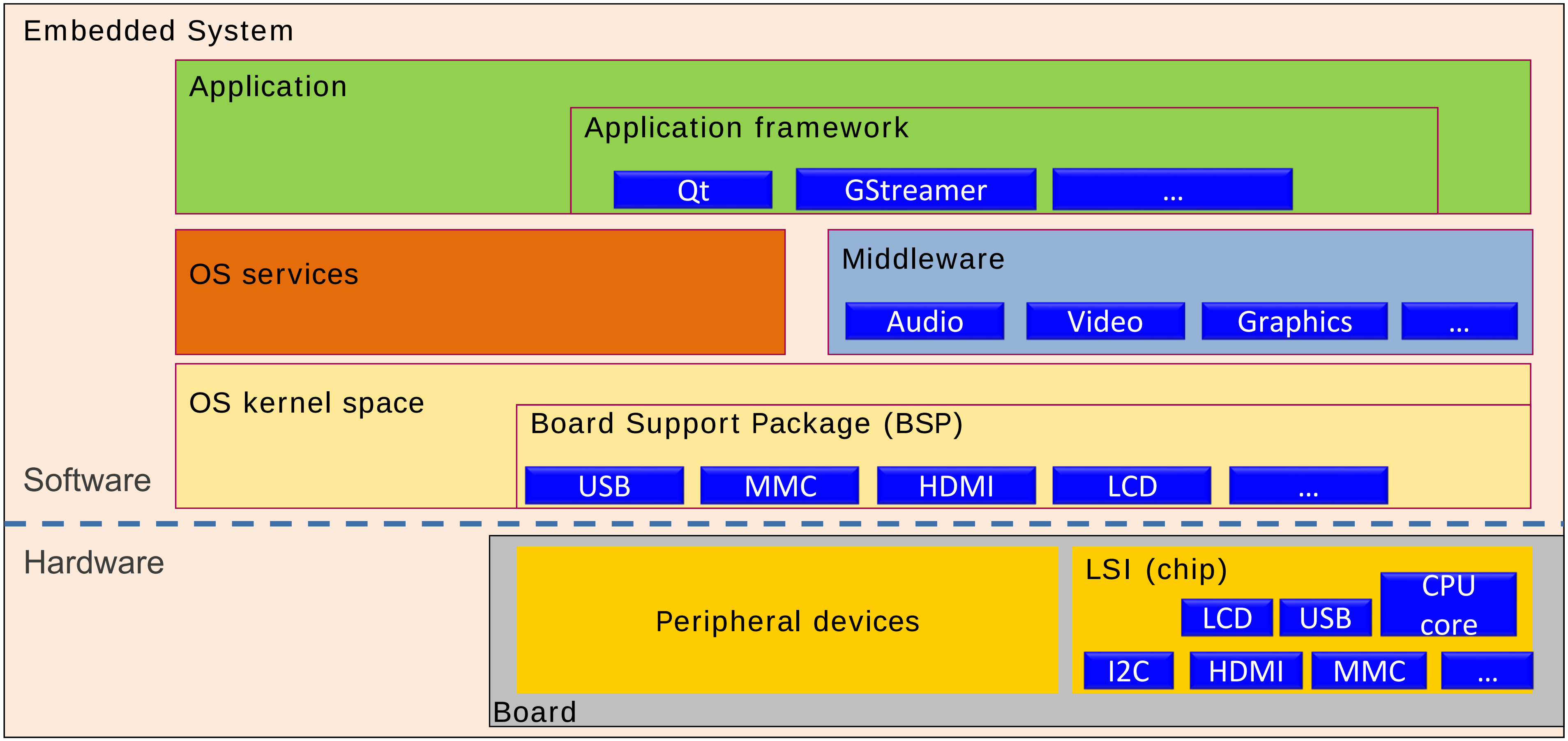


Software Structure (Typical Example)

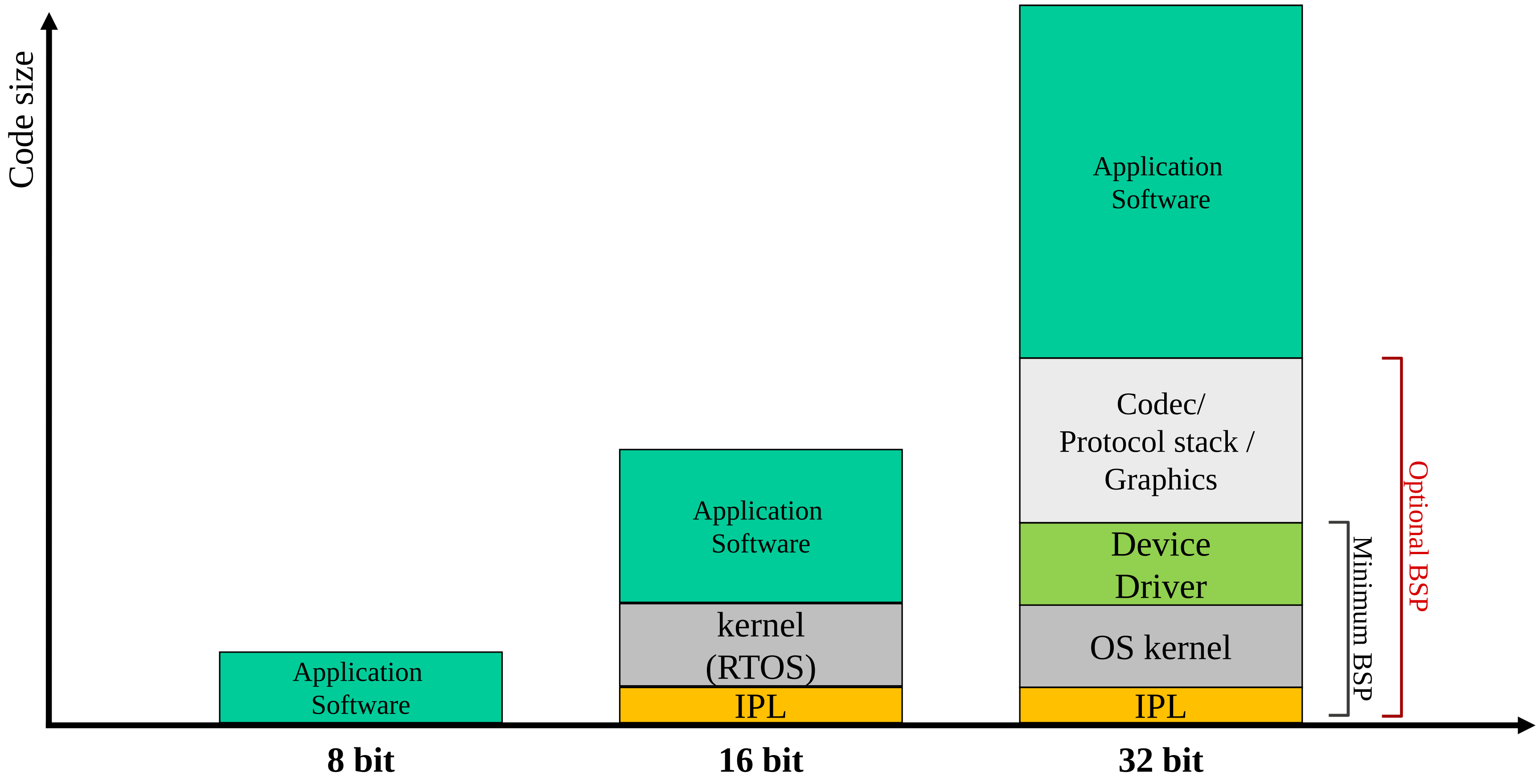
5. More Complex Software Structure



Architecture of an Embedded System

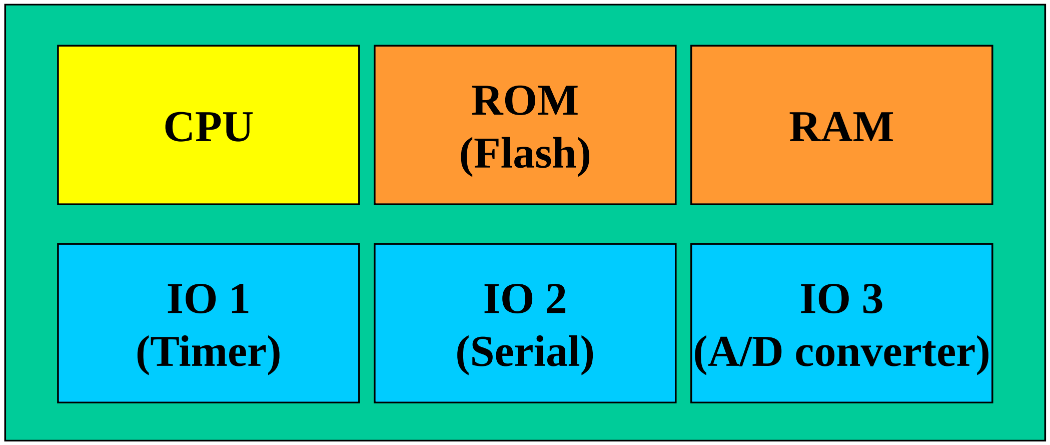


Code size and software structure examples



Type of Device – MCU and MPU

MCU: Micro Controller Unit (Or 'Single Chip Micro Computer')
All component is in one silicon ; CPU, memory (ROM and RAM), and IO

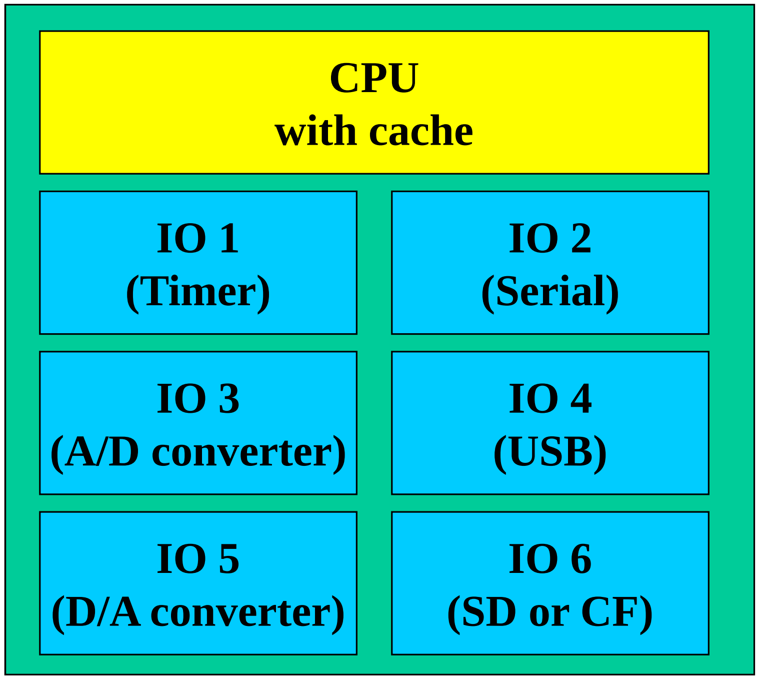


Smaller size of memory
Compact IO

MCU Device (Example)

Type of Device – MCU and MPU

MCU: Micro Processor Unit
CPU, cache memory and IO, external memory



CPU
with cache

IO 1
(Timer)

IO 2
(Serial)

IO 3
(A/D converter)

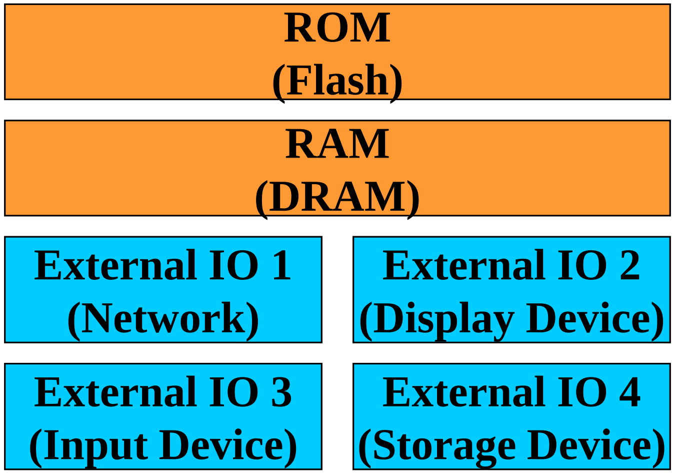
IO 4
(USB)

IO 5
(D/A converter)

IO 6
(SD or CF)

MPU Device (Example)

Larger size of memory
Rich IO



ROM
(Flash)

RAM
(DRAM)

External IO 1
(Network)

External IO 2
(Display Device)

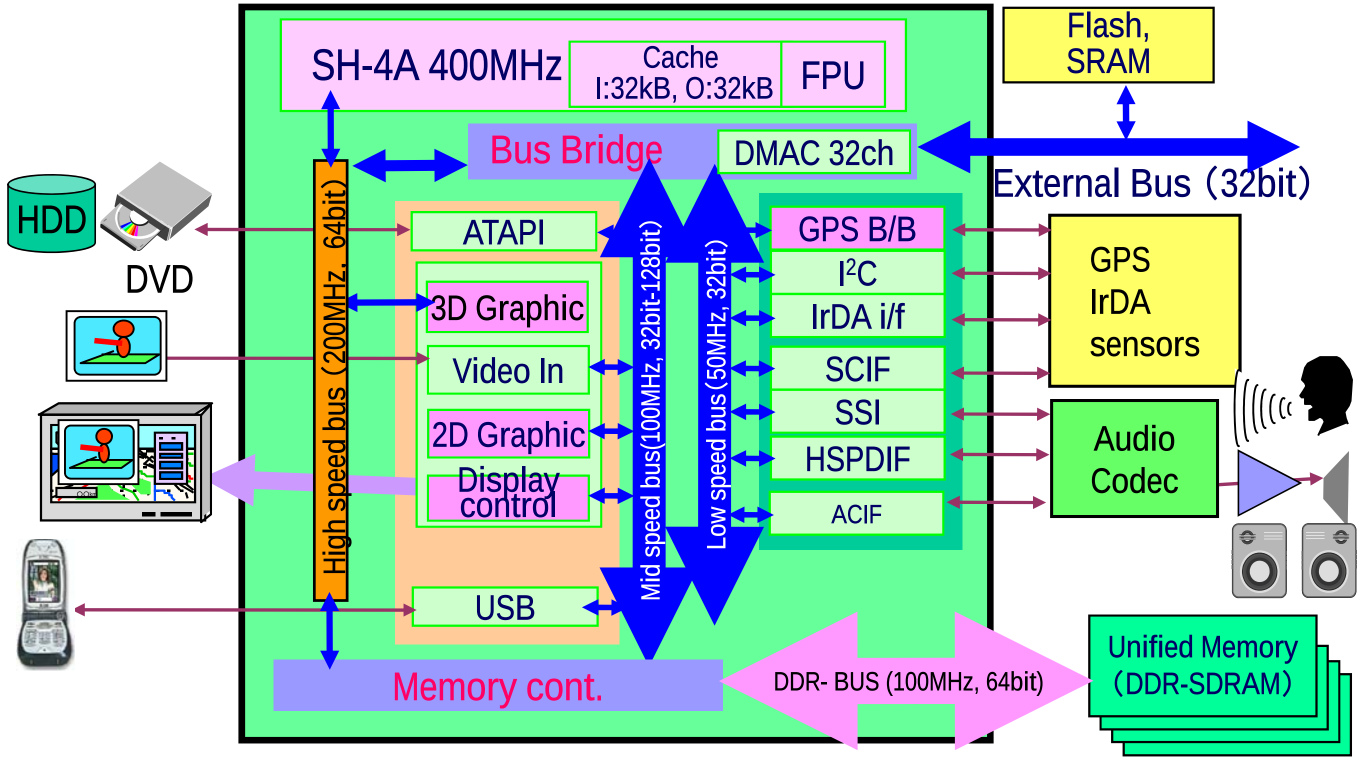
External IO 3
(Input Device)

External IO 4
(Storage Device)

MPU Device Example – SH-Navi I (SH7770)

- SH-4A CPU core (upward compatible from SH-4)
- Various IP module
- Effective Bus structure

Effective Usage of software asset
Unified Hardware platform approach
(from low to High end system)

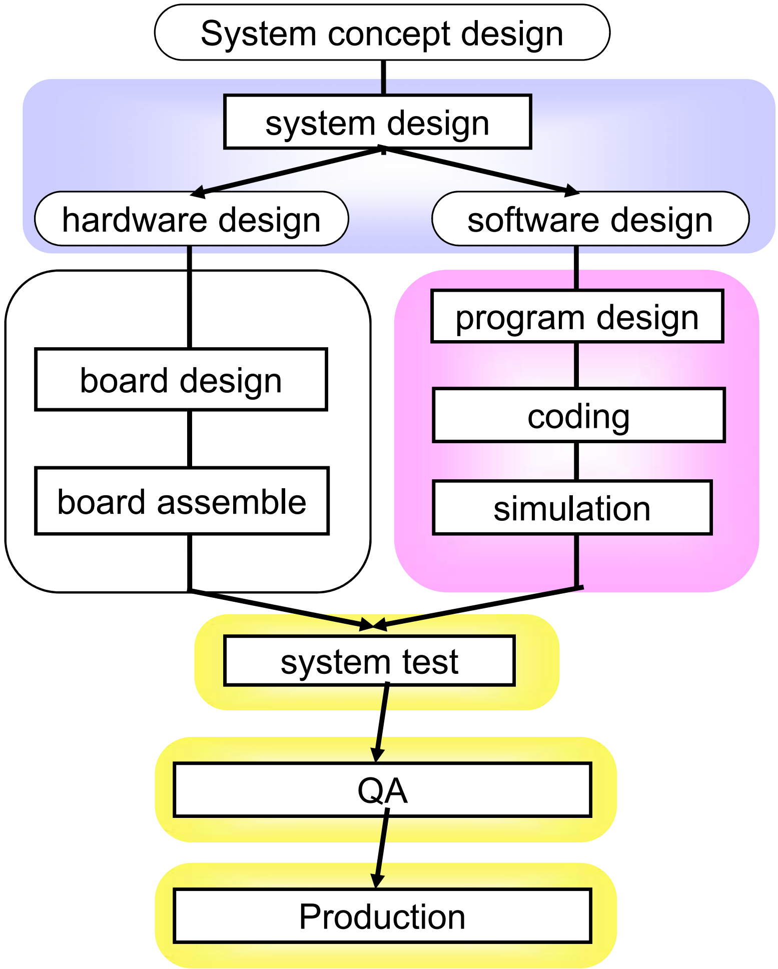


Embedded system development flow

Development flow

When you design the micro computer embedded system, you should design your system outline at first. And have to consider about……

- ➢How to reduce the product cost?
- ➢How to reduce the development cost?
- ➢How to improve the system quality?



System design target

When you design the micro computer embedded system, your targets are…

(1) How to reduce the product cost?

- o Choose the cost-competitive device as a Renesas micro processor!
- o Choose the valued parts to reduce the production cost.
- o Re-programmable device as micro controllers with on-chip flash memory.

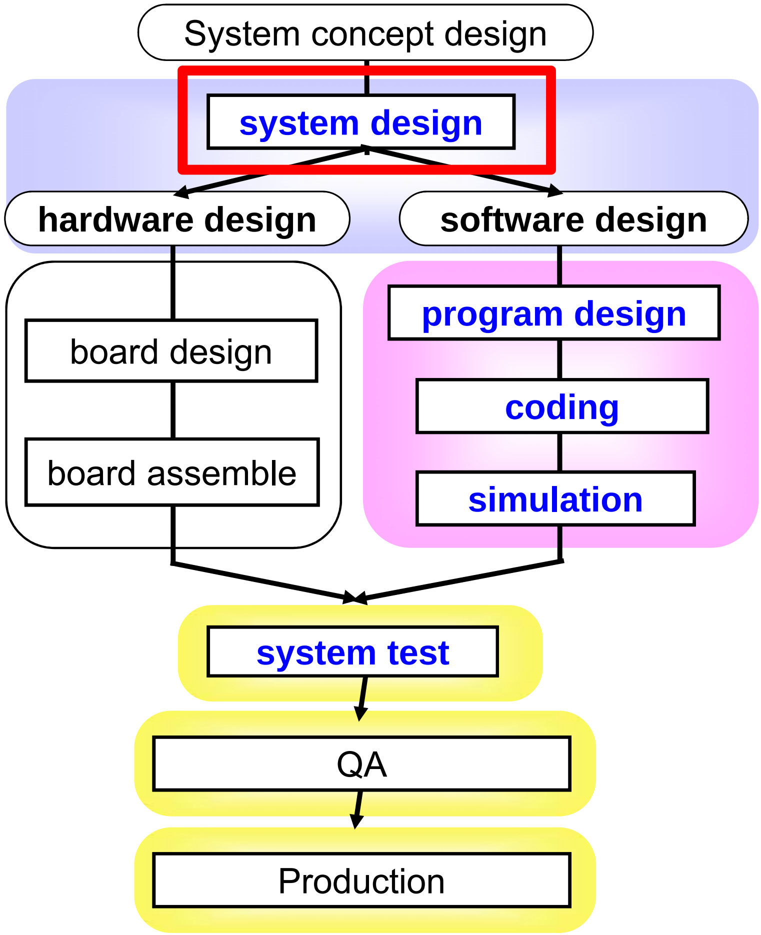
(2) How to reduce the development cost?

- o You should concentrate on your application. You do not design all of your system. Your system value is in your application!
- o Choose the proper operating system that is suitable for your system.
- o Choose the proper middleware that is suitable for your system.

(3) How to improve the system quality?

- o Choose the proper tool and method which conducts your system higher quality.
- o Choose proper debug tools that show the debug information on your demand.

Development tools help your system design

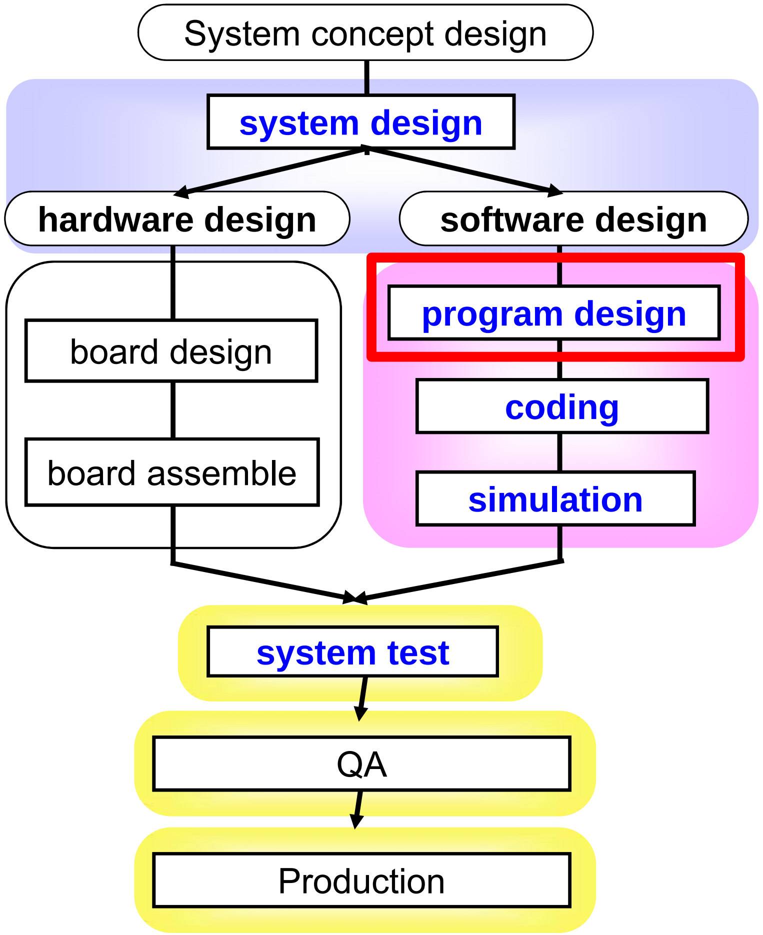


Not only function but also performance evaluation on reference platform is important.

- ✓ The reference platform is a board with microprocessor, memory and IOs.
- ✓ BSP is useful package for system / software evaluation.

T-Engine, an open and standardized platform, improves development efficiency of ubiquitous equipment.

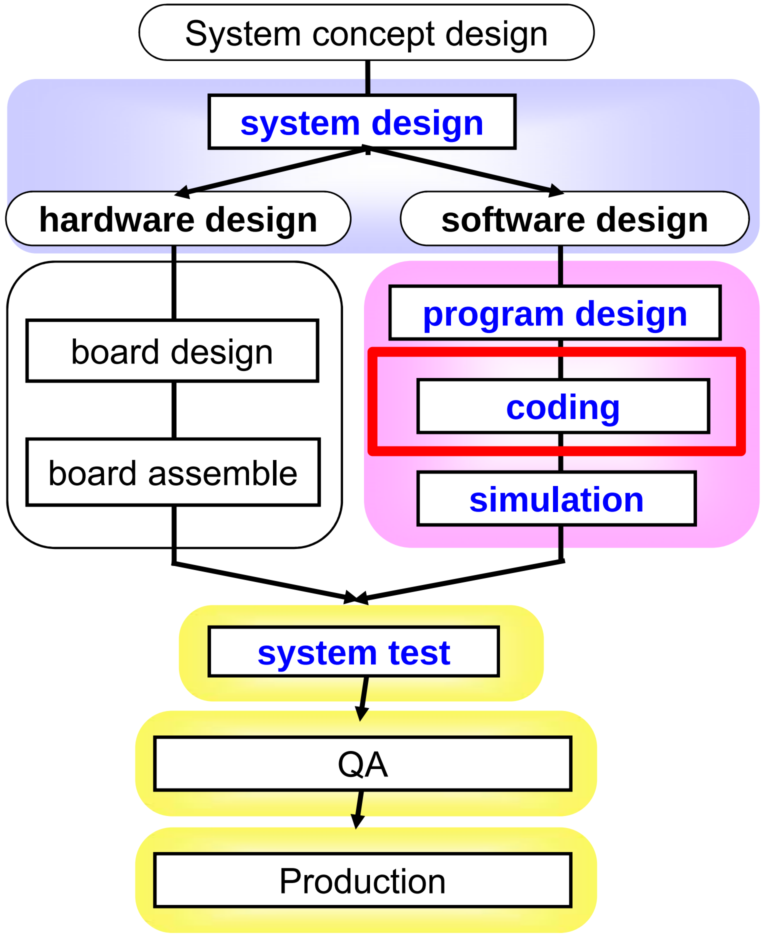
Development tools help your system design



When you start designing the software, it is possible to take proper software sub-component to reduce your software development cost and time, such as commercially available software package, legacy software in your previous product.

✓ If you will develop complex and large software system, OS (operating system) helps you to develop complex application quickly.

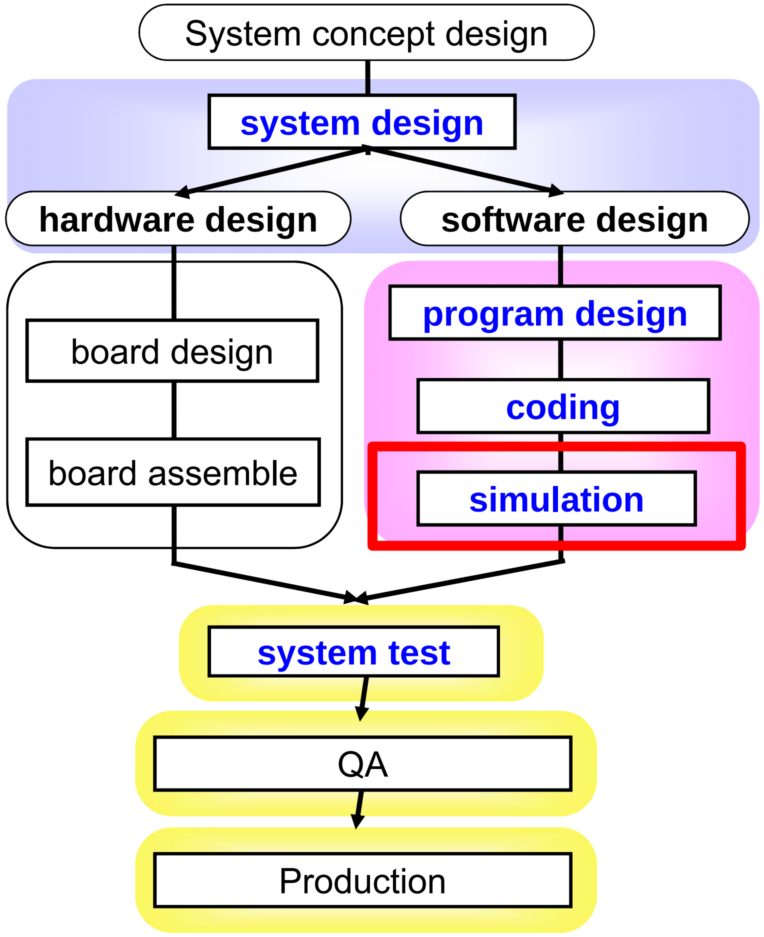
Development tools help your system design



You have to pay attention to reduce the code size for saving the memory space (in small memory size system), or to higher execution speed (performance), as hardware resources in embedded system is not rich (to reduce cost).

✓ C/C++ compiler provides you the code optimization to generate smaller and/or high performance code.

Development tools help your system design



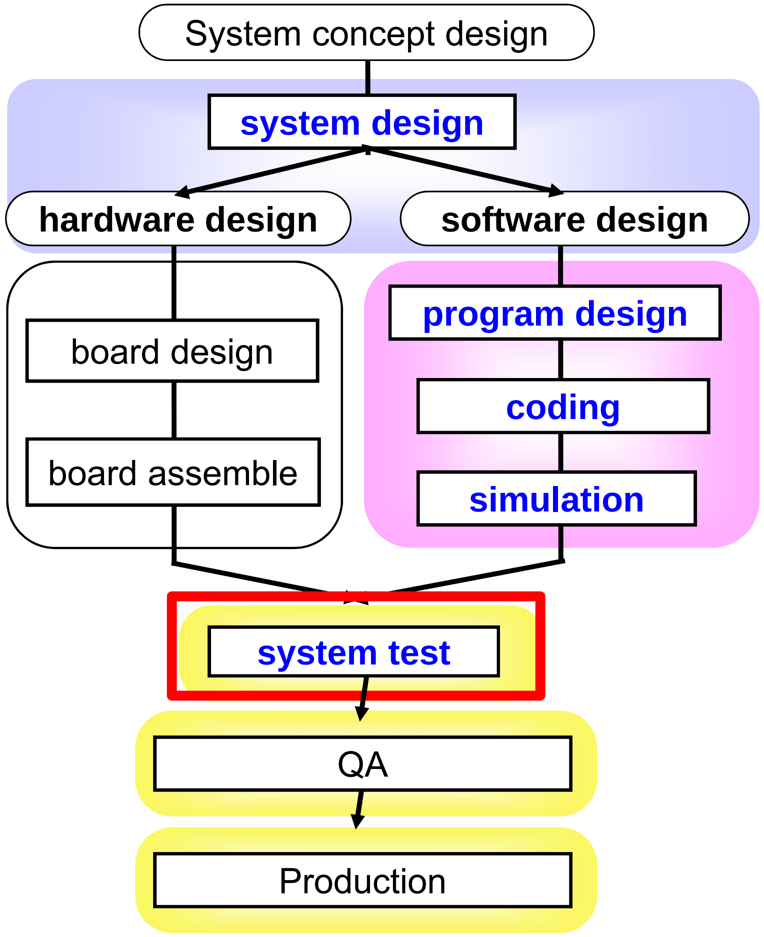
Simulator enables you to run your code on host PC (code is compiled for your target CPU, such as H8, M16C, SH-4, etc.)

It is better to test/debug your source code on simulator, to get better software quality.

Sometime, your target hardware is not available when you finished your coding, then simulator is only one way to test your code.

- ✓ Software must be tested in smaller unit first.
- ✓ Simulator gives you whole CPU resource view.
- ✓ Cycle accurate simulator gives you the certain software performance.

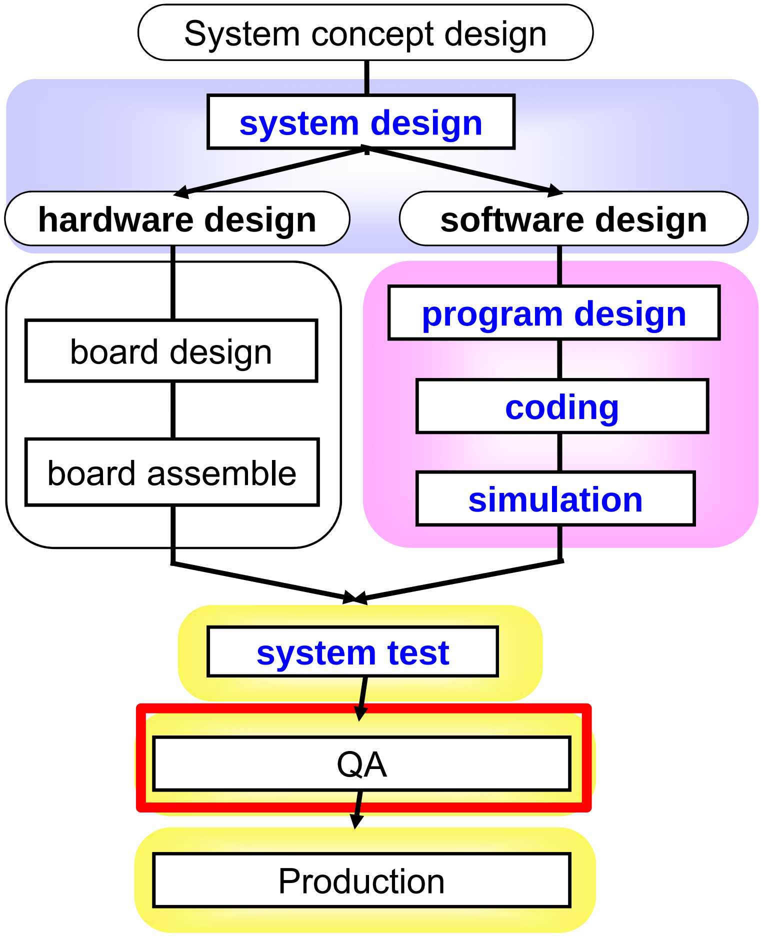
Development tools help your system design



When you finish software debug (and hardware debug), system test (run software on target hardware) must be done in properly.

- ✓ ICE (In-Circuit Emulator) gives you visibility of target hardware internal status, and gives you the control of target hardware.
- ✓ Basic hardware debug must be completed, before starting system test.

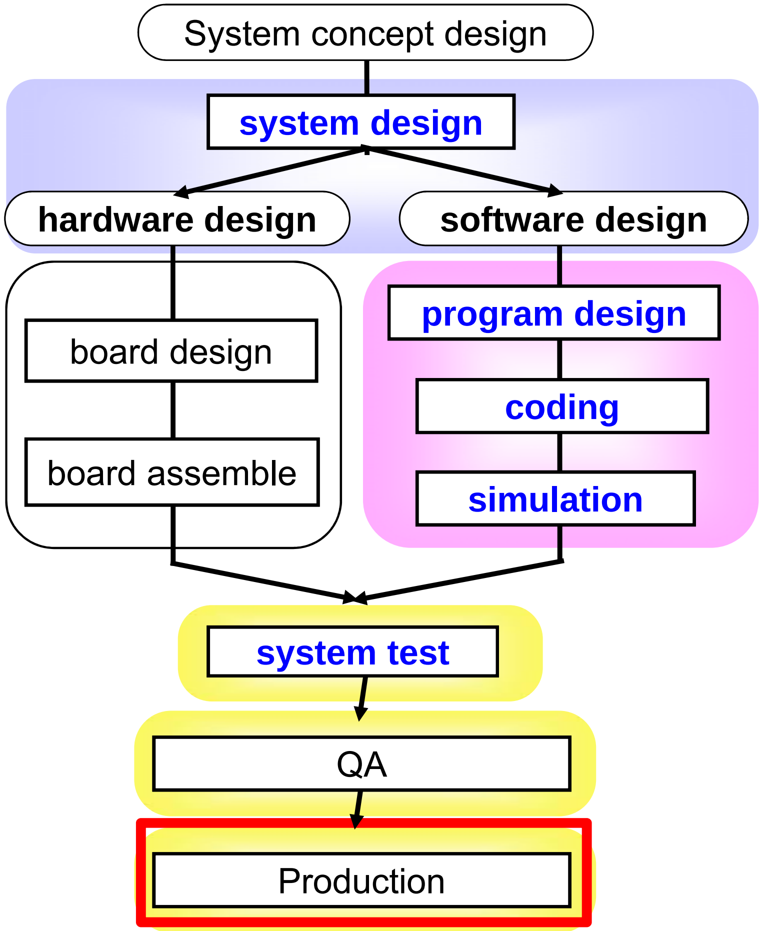
Development tools help your system design



You must define test scenario to get proper quality suitable for production.

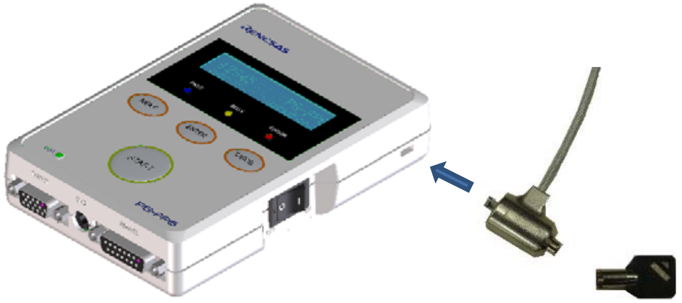
- ✓ Code coverage is one of software quality index in test phase.
- ✓ Code coverage information tells you how many instructions are tested (covered) in specific test suites against all instructions in your source code.
- ✓ One important test criteria for high quality software is to achieve 100% coverage.

Development tools help your system design



When your product will go into mass production, your code must be stored in target hardware (many number of hardware).
In recent embedded system, most object code is stored in flash memory.

- ✓ Flash programming tool helps you to transfer object code from host PC to target memory (especially for flash memory in MCU) in production.



RENESAS PG-FP6

Operating System

- ➢ Embedded Operating System Overview
- ➢ Windows Embedded CE
- ➢ Linux

Embedded Operating System Overview

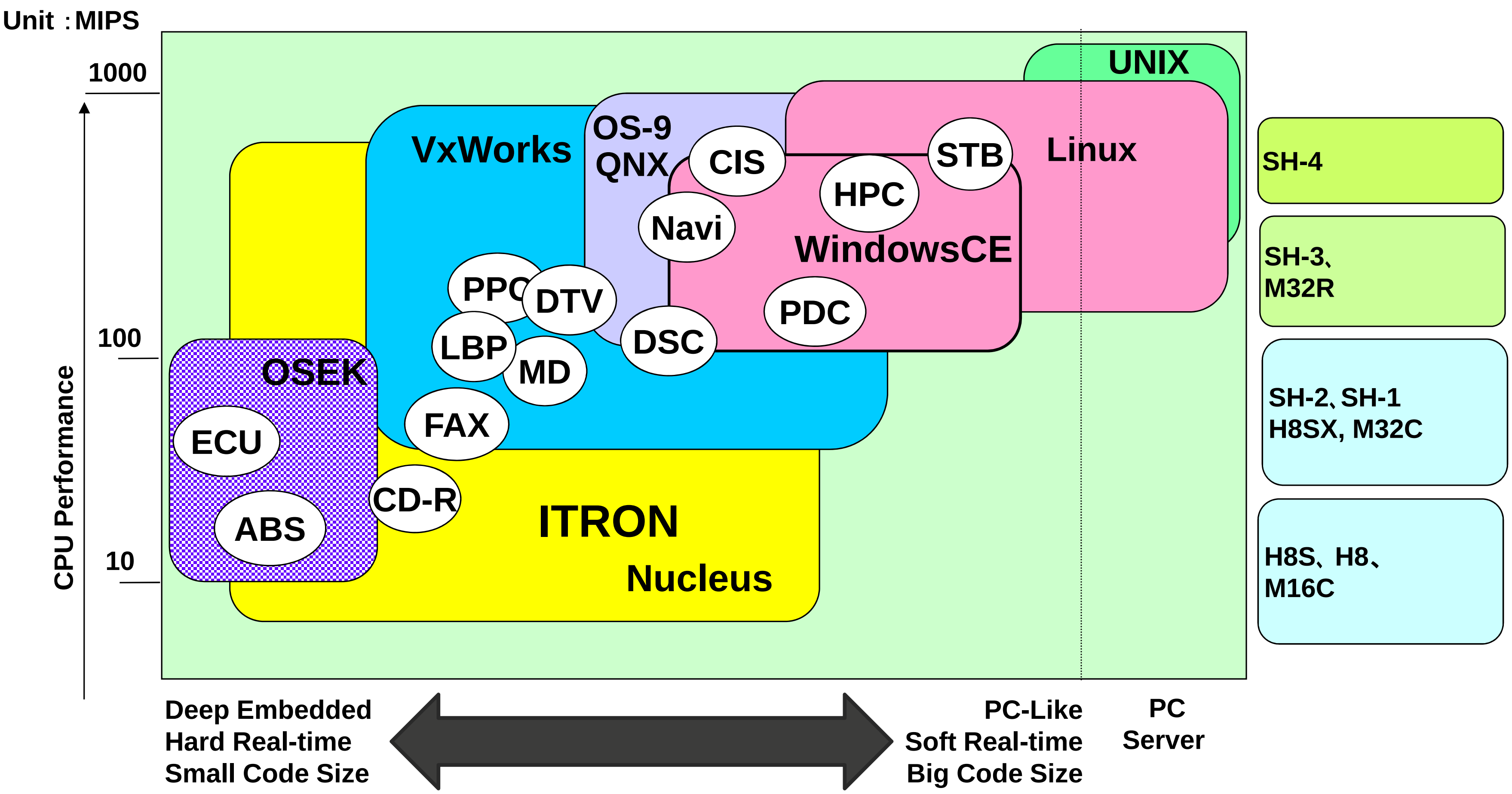
❑ Embedded Operating Systems for Renesas MCU and MPU

❑ Reasons for OS selection

❑ Kernel architecture difference

❑ Process, task and thread

Embedded Operating Systems for Renesas MCU and MPU



Criteria for OS selection

❑ Software richness: How many drivers, middlewares, protocols are provided/supported

❑ Foot print: Size of memory.

❑ Realtimeness, performance: Response latency against interrupts, speed of task switch (process re-scheduling), etc.

❑ Tool chain: Not only compiler/debugger, but also other useful tools, such as system configuration tool, performance analyzer, etc.

❑ Robustness and security: MMU protection, kernel/driver stability; Secure protocol support

❑ Technical support, BSP support, maintenance, update service

❑ Easiness: Easy to port, easy to develop

❑ Price, Legal (GPL, patents, 3rd party intellectual property rights, etc.)

❑ Others : There might be many other points

What is OS major function

Resource management

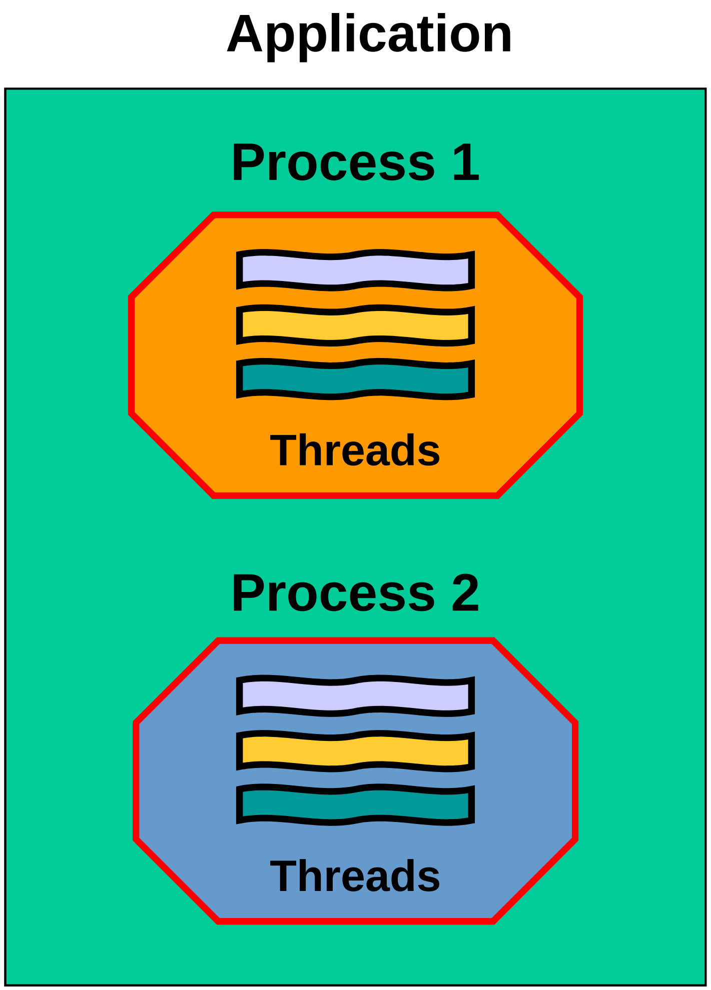
Each process/task uses memory and IO resources independently. OS has to manage resource allocation properly, otherwise resource conflict makes fatal error on the system.

Scheduling

Each process/task must be executed in proper order, to achieve low cost/high performance embedded system.

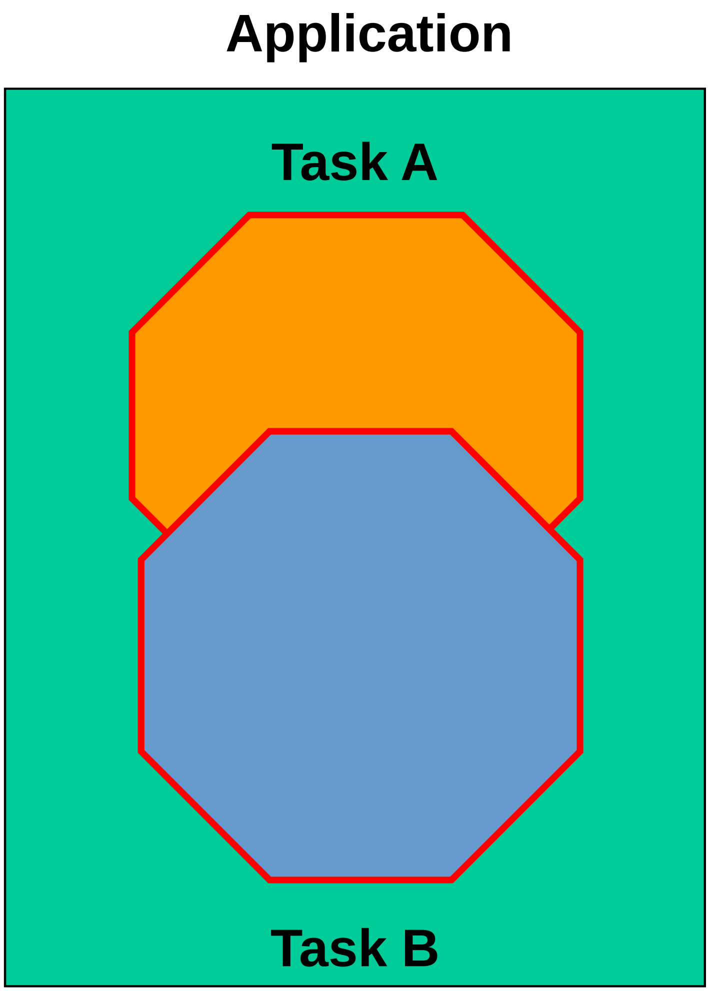
Process and thread

- ❑ Application is divided into memory protected units called processes
- ❑ Process is further divided into internal, schedulable units called threads
- ❑ Threads share all of the same resources (memory space included)



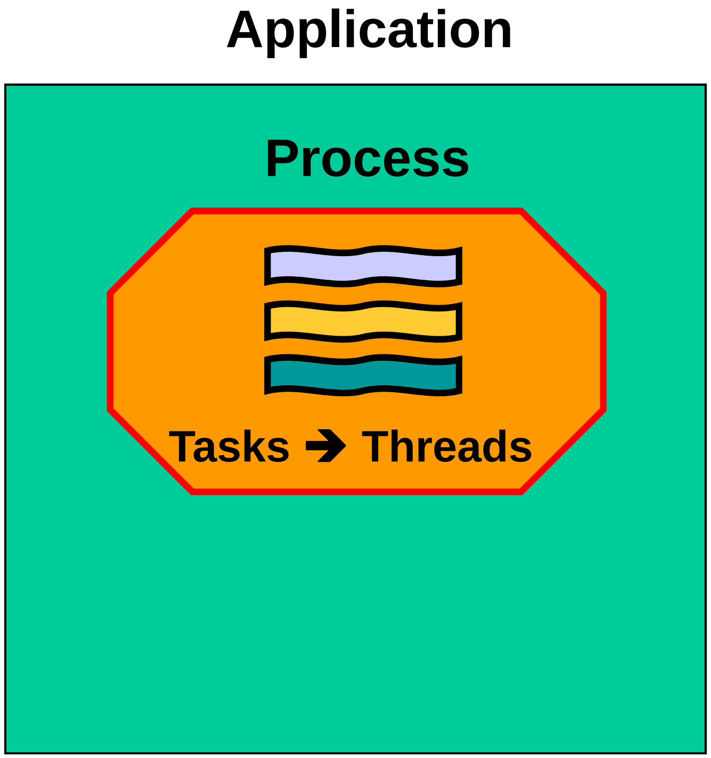
Task

- ❑ Application is divided into non memory protected units called task
- ❑ Tasks share all of the same resources (memory space included)



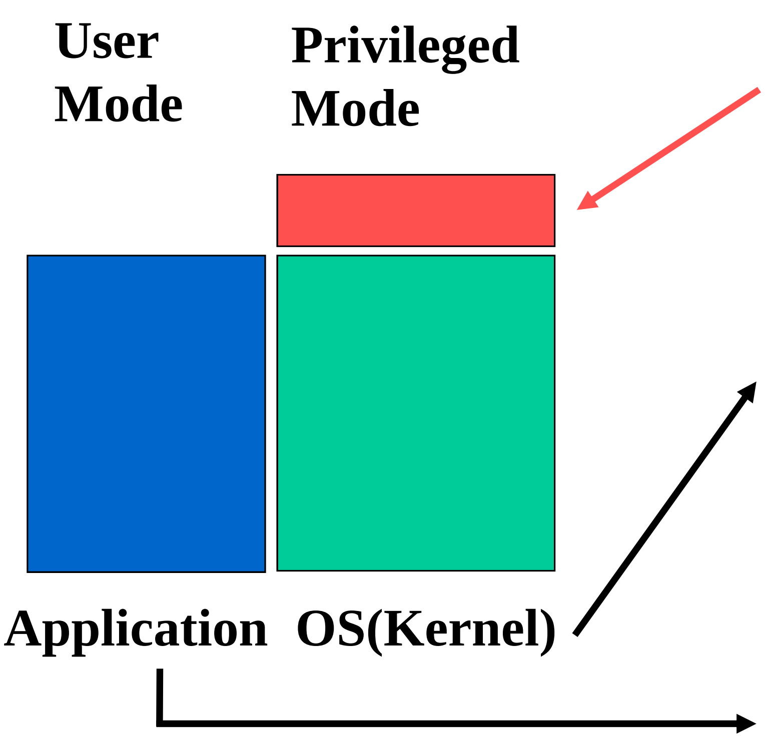
Process and thread

- ❑ Application porting from real-time kernel based system to Linux based one, such as ITRON → Linux
  - o To share resources, each task is mapped as thread
  - o One process in one application
  - o Application itself may run as designer expected, but this implementation is not based on original process idea
    (Unix POSIX interface)
  - o Performance (or other) problem may occur on this kind of migration



Programming model for High-end MPU

Programming model for SH-4, generic



User Mode

Privileged Mode

Application

OS(Kernel)

Specific Function for OS,
only available in Privileged Mode

Run in privileged mode, MMU off,
no protection within privileged mode,
OS kernel is protected from application
in User Mode

Run in user mode, MMU on,
Process is protected each other

MMU

MMU (with OS) provides memory protection and dynamic loading by following function:

❑ Address translation, from logical (virtual) address to physical address
→ dynamic loading, physical address extension beyond 32 bit (32 bit CPU)

❑ Address translation is done in small address size unit called page, typical page size is 4 kByte/page

❑ Memory access type setting for page entry, RO/RW/EX, PV/US
→ protect the access beyond process,
protect the kernel from illegal access caused by application

RO : Read Only, RW : Read and Write, EX : Execute only
PV : Privileged mode, US : User mode

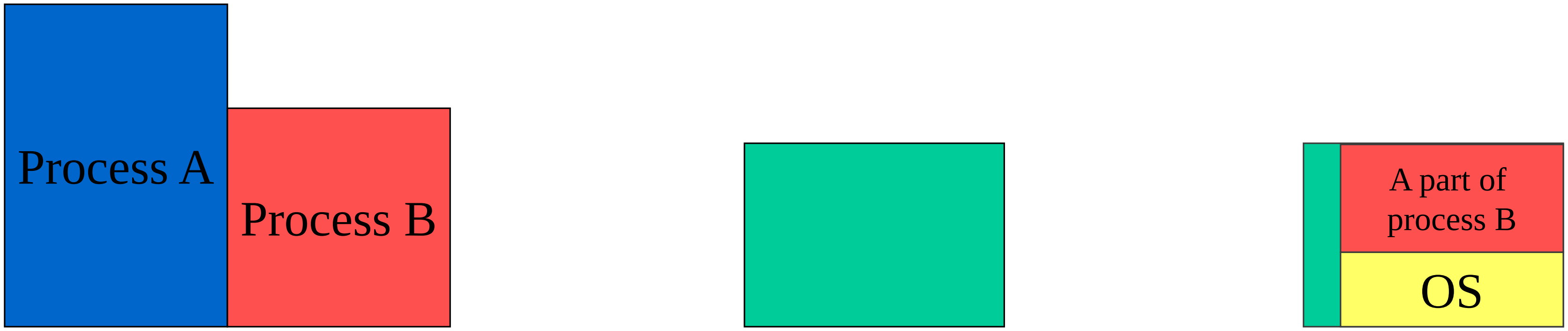
Example of memory mapping

Traditional model
(memory is smaller than process in application)

Application
(Logical Address)

Memory
(Physical address)

Application allocation
in memory



Process A

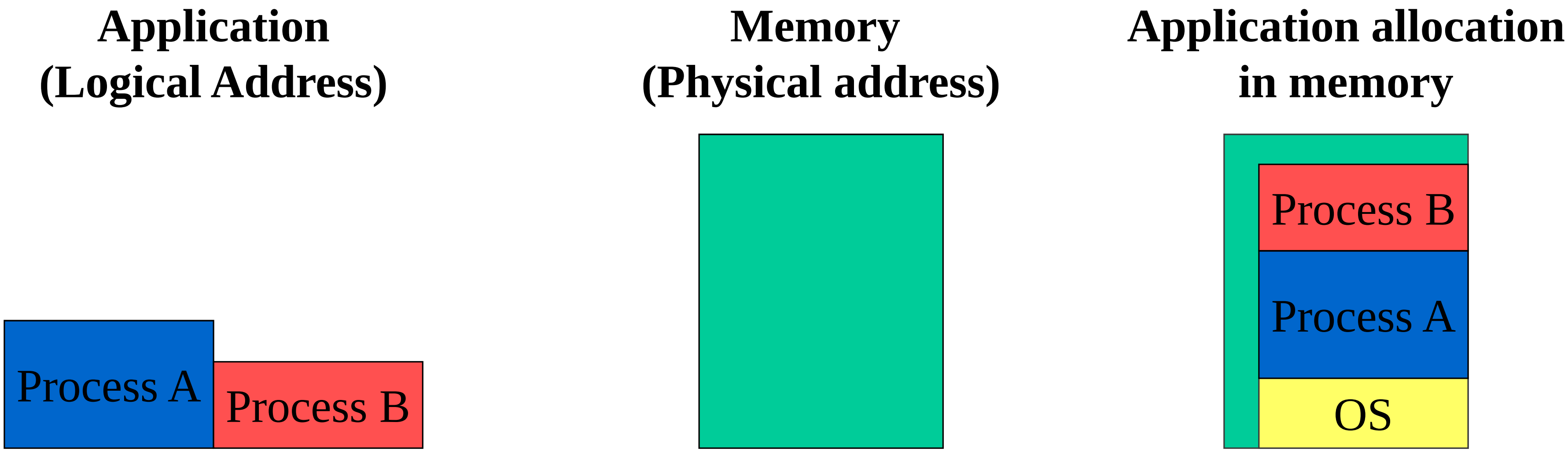
Process B

A part of
process B

OS

Example of memory mapping

Memory model in modern embedded RTOS
(memory is large enough to place multiple process)



Application (Logical Address)

Process A

Process B

Memory (Physical address)

Application allocation in memory

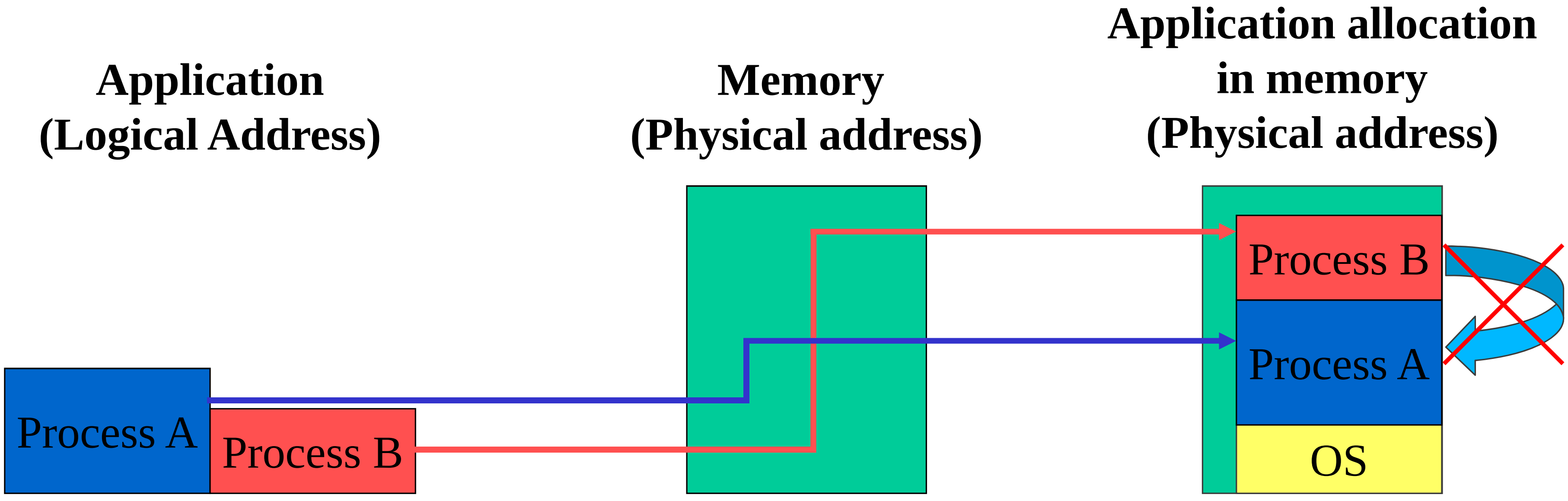
Process B

Process A

OS

Address translation by MMU

MMU will decide proper physical address for each application and OS,
Physical address is not same as Logical address in most cases.
Processes are protected thoroughly.



Access beyond process is NOT allowed

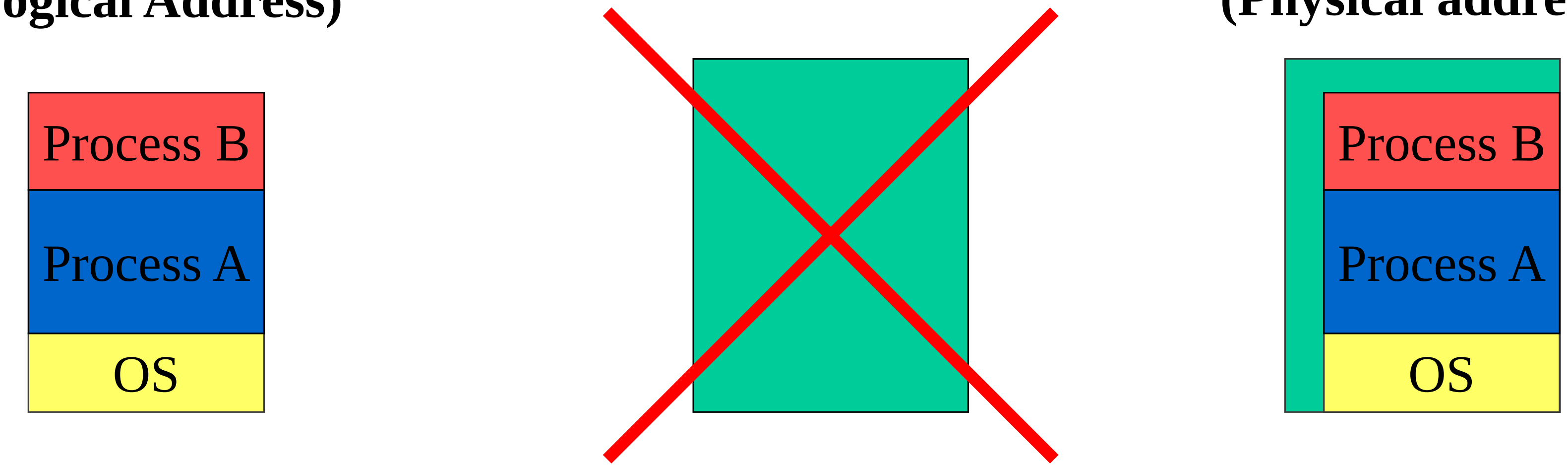
Address allocation in non-MMU device

Application software engineer have to decide proper physical address for each application and OS, when he/her compile and link the program. Logical address must be same as physical address.

Application (Logical Address)

No MMU

Application allocation in memory (Physical address)



Process B

Process A

OS

Process B

Process A

OS

Dynamic Link and Static Link

❑ with MMU

- Application software engineer does not need to care about physical address. Just compile, link and go.
- OS takes care of physical address resolution. OS places each process into proper physical address through MMU. → Dynamic link

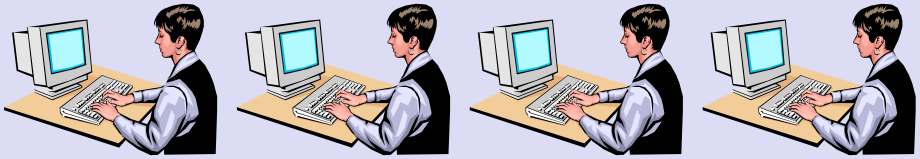
❑ without MMU

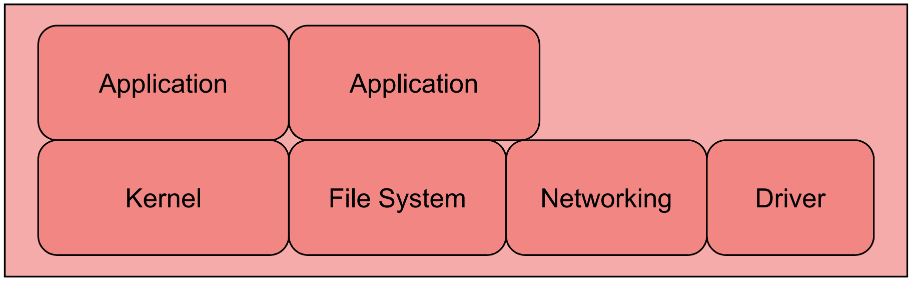
- Application software engineer have to think about physical address for his/her application.
- Application software engineer have to specify correct physical address on compile and link. → Static link

Kernel architecture example

Real-time kernel without MMU protection

- **No MMU protection**
- **Application, driver, protocols are in kernel space**





Application

Application

Kernel

File System

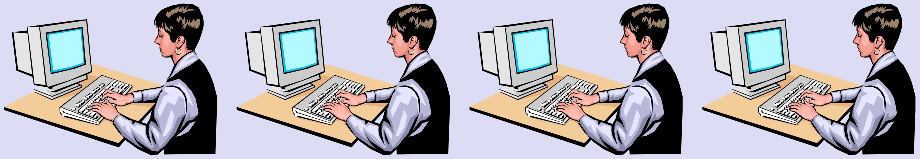
Networking

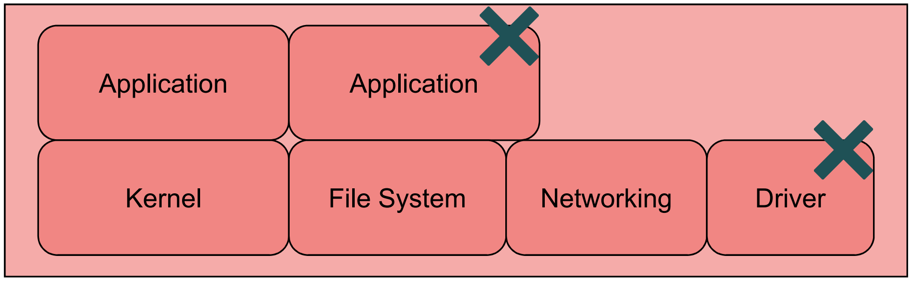
Driver

Kernel architecture example

Real-time kernel without MMU protection

- **No MMU protection**
- **Application, driver, protocols are in kernel space**

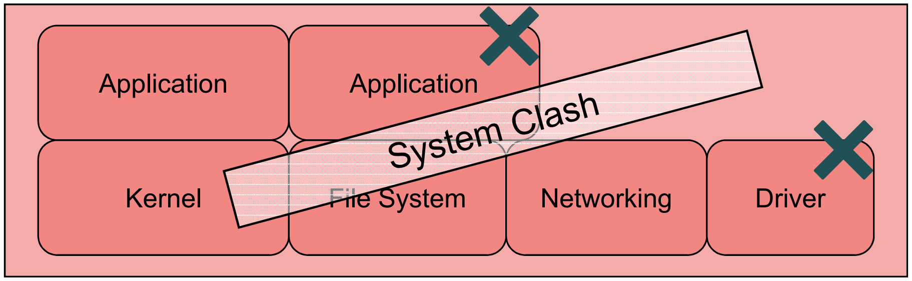




Kernel architecture example

Real-time kernel without MMU protection

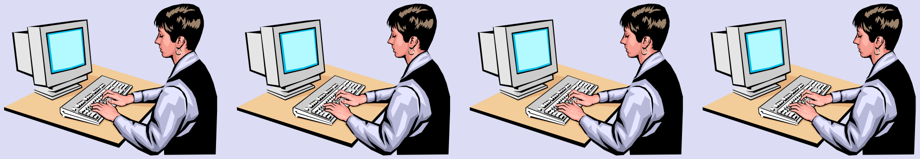
- **No MMU protection**
- **Application, driver, protocols are in kernel space**

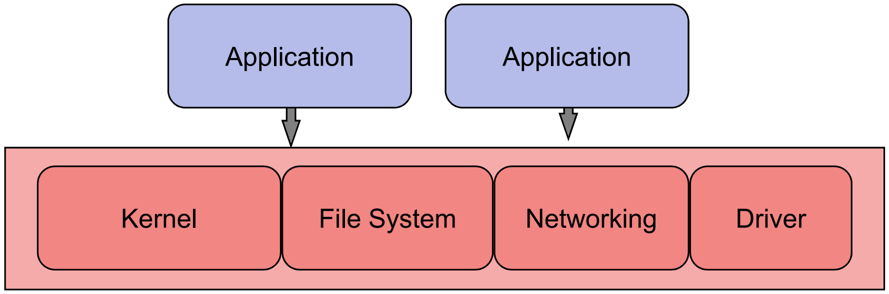


Kernel architecture example

Monolithic kernel (NT / Unix / Linux)

- **Partial MMU protection**
- **Application is protected**

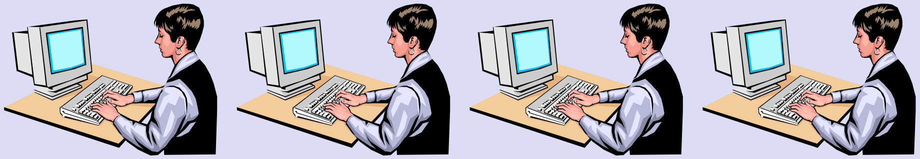


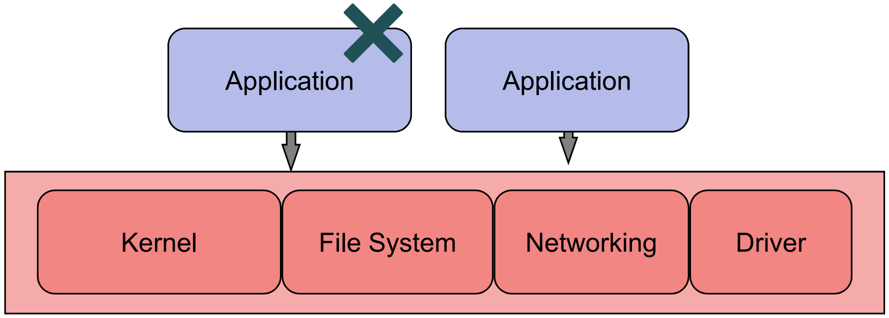


Kernel architecture example

Monolithic kernel (NT / Unix / Linux)

- **Partial MMU protection**
- **Application is protected**

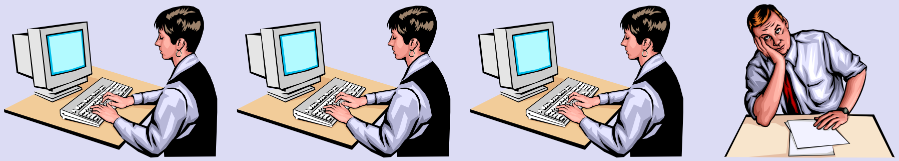


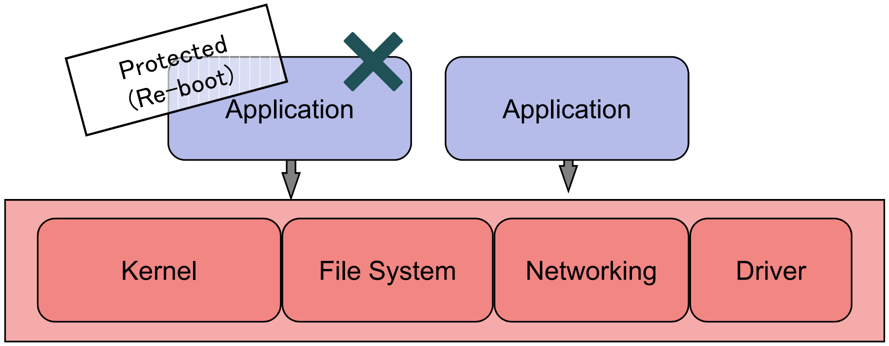


Kernel architecture example

Monolithic kernel (NT / Unix / Linux)

- **Partial MMU protection**
- **Application is protected**

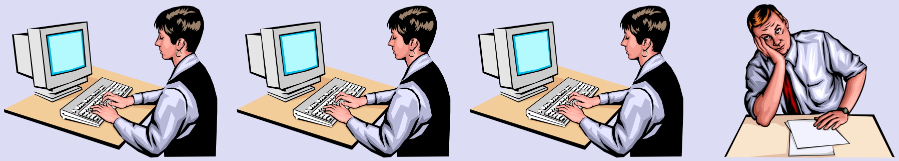


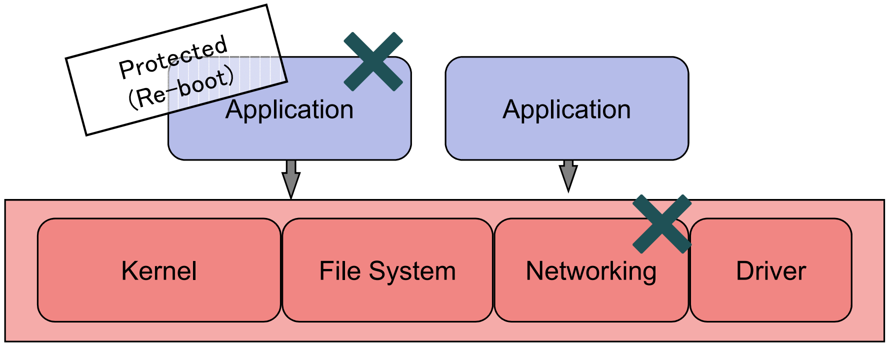


Kernel architecture example

Monolithic kernel (NT / Unix / Linux)

- **Partial MMU protection**
- **Application is protected**

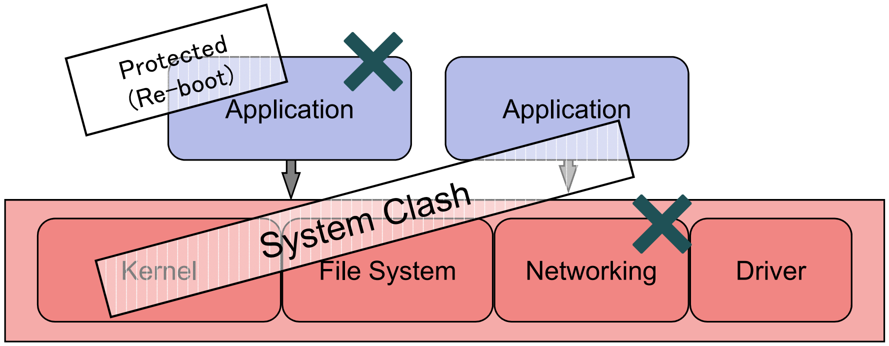




Kernel architecture example

Monolithic kernel (NT / Unix / Linux)

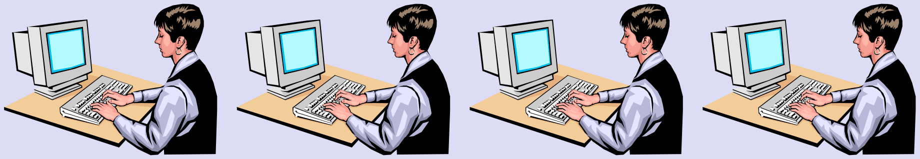
- **Partial MMU protection**
- **Application is protected**

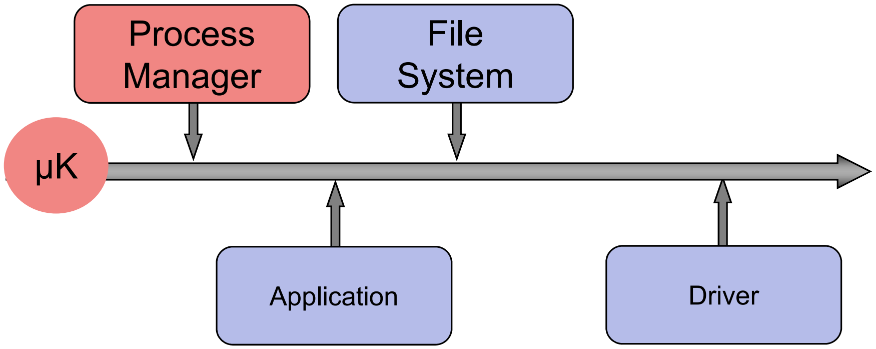


Kernel architecture example

Real-time kernel with MMU protection

- **Application, Driver, protocols are protected**





Process Manager

File System

µK

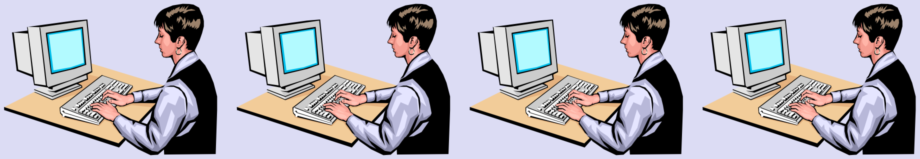
Application

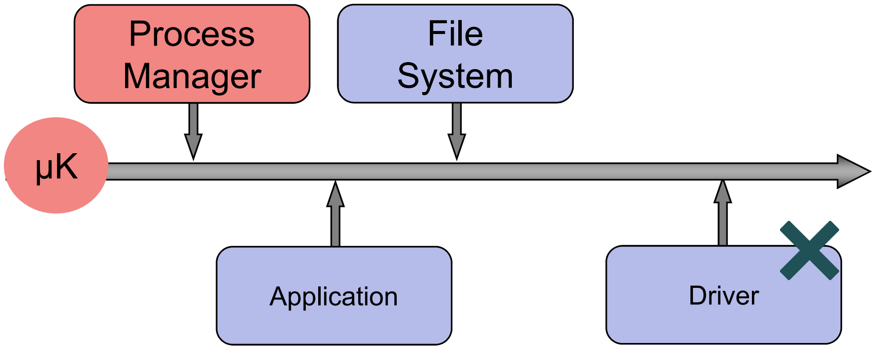
Driver

Kernel architecture example

Real-time kernel with MMU protection

- **Application, Driver, protocols are protected**

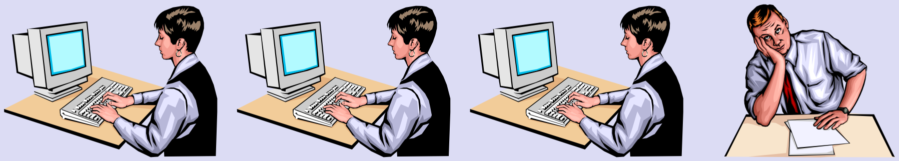


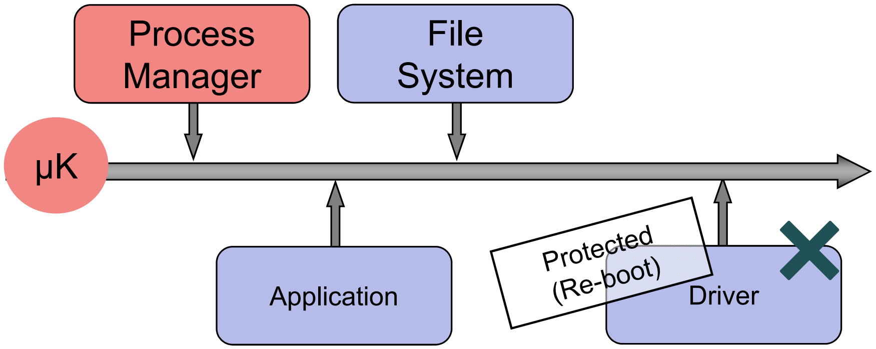


Kernel architecture example

Real-time kernel with MMU protection

- **Application, Driver, protocols are protected**

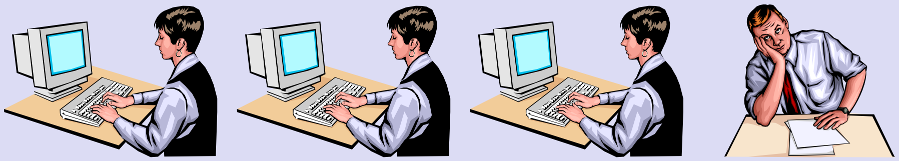


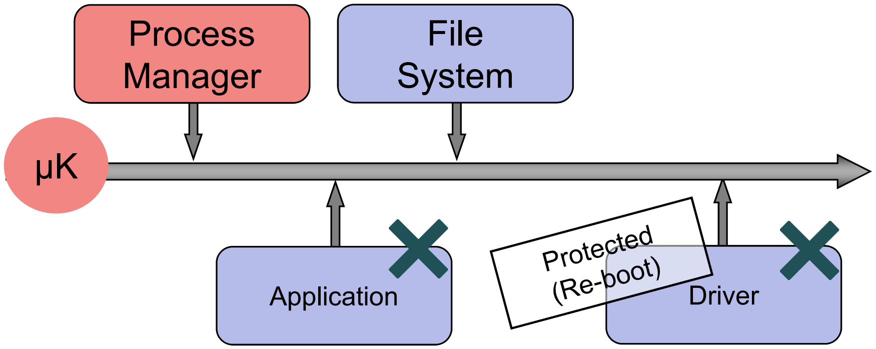


Kernel architecture example

Real-time kernel with MMU protection

- **Application, Driver, protocols are protected**

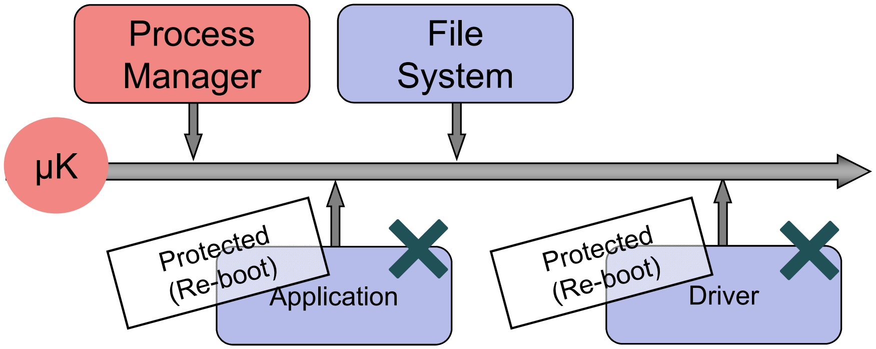




Kernel architecture example

Real-time kernel with MMU protection

- **Application, Driver, protocols are protected**



Kernel architecture example

❑ Compact real-time kernel (real-time OS) vs. OS (with MMU protection)

- o Small size vs. larger size
- o No MMU protection vs. MMU protection
- o Fast vs. slow
- o Poor driver/protocols vs. rich driver/protocols
- o Lower price vs. higher price
- o Reliability, stability is also need to be considered
- o OS (or kernel) specification is different each by each, above classification is example for study purpose.

Windows CE

- ❑ History
- ❑ Windows Embedded CE 6.0 overview
- ❑ Development Environment and debug

Windows CE – History

□ Windows CE 1.0	1996.11	Pegasus
□ Windows CE 2.0	1997. 9	Alder
□ Windows CE 3.0	2000. 4	Cedar
□ Windows CE .NET 4.0	2002. 1	Talisker
□ Windows CE .NET 5.0	2004. 7	Macallan
□ Windows Embedded CE 6.0	2006. 9	Yamazaki
□ ...		

Windows CE – Application specific package

- ❑ Windows Automotive (WA)   V. 5.0 / V5.5 (2008)
  - o Developed by Microsoft Japan. Widely used in Japan, and Asia
- ❑ Microsoft Auto (MS Auto) V. 3.0 (2008) / V. 4.0 (2009)
  - o Developed by Microsoft US. Used in Europe, North America
- ❑ New OS for Automotive (2010)
  - o Based on WinCE V7 (SMP capable)
  - o Merged release of MS Auto and WA
  - o Multi-core (SMP) and single-core support
  - o Silverlight technology based new graphics system and tools
  - o Automotive System Tool (Readyguard, Snap Shot Boot, etc.)
  - o Mobile and multimedia capability inherited from MS Auto

Windows CE – Feature

- ❑ Win32 API
- ❑ New kernel : up to 2GB address space, up to 32000 process
  - o High performance
  - o High Reliability
- ❑ Supports 4 CPU architectures : x86, SH-4, ARM, MiPS
- ❑ Microsoft quality
- ❑ Network media device support
- ❑ DVR Engine : multiple digital video stream support

MiPS: Microprocessor without Interlocked Pipeline Stages

出典：「Windows Embedded CE 6.0 組み込みOS構築技法入門」
日経BPソフトプレス社刊

Windows CE – Development environment

- ❑ Host Machine : Windows 2000, Windows XP, or Windows Vista
- ❑ Microsoft Visual Studio 2005
- ❑ Windows Embedded CE 6.0 Platform Builder
- ❑ Additional software
    - o Install BSP, etc., for specific device/platform support

Windows CE – Debug

❑ KITL (Kernel Independent Transport Layer)

- o Connection interface between target hardware and Host PC (Platform Builder)
- o Serial or Ethernet connection

❑ Debug function on Platform Builder

- o Kernel debugger
- o Remote tool
- o Target control (command line for debug)

❑ eXDI (eXtended Debug Interface)

- o hardware assist (by ICE) debug function

出典： 「Windows Embedded CE 6.0 組み込みOS構築技法入門」
日経BPソフトプレス社刊

Linux

❑ Why Linux? What is Linux?

❑ Linux vs. embedded OS

❑ Porting Linux from x86 to SH

❑ Linux package

❑ Open source vs. commercial distribution

❑ Embedded Linux

❑ GPL

- What is ‘free’?
- What is ‘open source’?
- What is ‘as is’?

Why Linux?

❑ Network, file system are supported

❑ High compatibility between CPUs

❑ Many peripherals, protocols are supported

❑ Many existing software source code files are available

❑ Relatively easy to build prototype system

❑ Open source

❑ Free of charge

What Linux?

- ❑ The word of ‘Linux’ has several different meanings
  - o Linux OS kernel, just kernel only
  - o Linux OS kernel binary, including drivers, etc.
  - o Linux command, such as bash
  - o Open source applications, such as samba, apache, running on Linux
  - o GNU development environment for Linux
  - o Linux distributions
  - o Combination of above

What Linux?

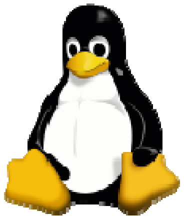
- ❑ The word of ‘Linux’ has several different meanings
  - o Linux OS kernel, just kernel only
  - o Linux OS kernel binary, including drivers, etc.
  - o Linux command, such as bash
  - o Open source applications, such as samba, apache, running on Linux
  - o GNU development environment for Linux
  - o Linux distributions
  - o Combination of above

What Linux?

❑ The word of ‘Linux’ has several different meanings

- o Linux OS kernel, just kernel only
- o Linux OS kernel binary, including drivers, etc.
- o Linux command, such as bash
- o Open source applications, such as samba, apache, running on Linux
- o GNU development environment for Linux
- o Linux distributions
- o Combination of above

The Linux kernel was conceived and created in 1991 by Linus Torvalds.
https://en.wikipedia.org/wiki/Linux_kernel



Tux the penguin, mascot of Linux

What Linux?

- ❑ Linux distributions (aka. distro)
  - o https://en.wikipedia.org/wiki/List_of_Linux_distributions
  - o A distro is an operating system made from a software collection, which is based upon the Linux kernel and, often, a package management system.
  - o Linux users usually obtain their operating system by downloading one of the Linux distributions.

What Linux?

❑ Linux distributions (aka. distro)

- https://en.wikipedia.org/wiki/List_of_Linux_distributions
- A distro is an operating system made from a software collection, which is based upon the Linux kernel and, often, a package management system.
- Linux users usually obtain their operating system by downloading one of the Linux distributions.
- Widely used distributions:

Linux vs. traditional embedded RTOS

Pros

- Wide range source code availability. Everyone has access to whole source code through internet
- Copy left: everyone can use it, no limitation for using Linux
- Free of charge (or seems to be free of charge)

Cons

- Kernel structure design concept, best match for PC, but…
  - Linux is TSS (Time Sharing System) based architecture
  - Slow latency against interrupts, process/task switch, etc.
  - Designed for x86 based PC architecture, not for embedded application
- Specification change / incompatibility
  - Linux specification is changing rapidly. Upward compatibility is not guaranteed. I.e., if you port some software onto another kernel, you must test it. Compatibility (that software should work on another kernel) is not guaranteed.
- Quality, guarantee
  - Linux quality is getting better, but no one provides formal guarantee
- GPL

Linux package components

- ❑ Kernel (vmlinux)
  - o Driver binary is statically linked to kernel
  - o Running in privileged mode on SuperH architecture (SH-4)

- ❑ RootFS (Root File System, or called ‘User Land’)
  - o On PC Linux, rootFS is placed on HDD. In embedded application, flash file system is popular than HDD file system
  - o Running in user mode on SuperH architecture (SH-4)

- ❑ Tool chain (gcc, binutils, glibc, etc.)
  - o Cross (on x86 Linux) and native (on SH-4 Linux) tool chain are available.
  - o Note that some gcc distribution includes newlib, not glibc, such as KPIT SH GCC.

- ❑ IPL (Initial Program Loader)
  - o Boot up, initialization of the system, loading kernel, etc.

Open source vs. commercial distribution

❑ Open source (kernel.org, fsf.org, etc.)

- o Free of charge
- o No warranty in any kind, user’s own risk

❑ Commercial distribution (Red Hat Enterprise Linux, Montavista Linux, etc.)

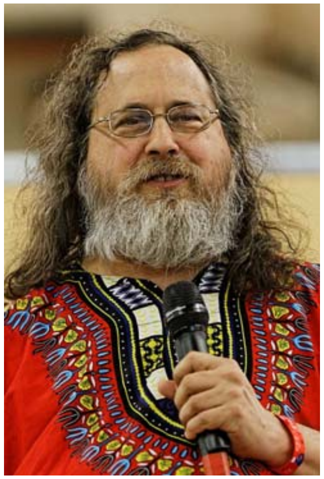
- o NOT free of charge
- o Limited warranty (depends on contract/price)
- o Bug fix, update, consulting services are available

GNU GPL (GNU General Public License)

- ❑ Most of Linux software is covered by GNU GPL (GPL v2/v3) license
- ❑ As Linux kernel binary file includes driver binaries, driver is also covered by GPL, even if the driver is coded from scratch by yourself.
- ❑ Please refer original GPL carefully at http://www.gnu.org/copyleft/gpl.html, when you are in charge for Linux based software development

GNU GPL (GNU General Public License)

- ❑ Most of Linux software is covered by GNU GPL (GPL v2/v3) license
- ❑ As Linux kernel binary file includes driver binaries, driver is also covered by GPL, even if the driver is coded from scratch by yourself.
- ❑ Please refer original GPL carefully at http://www.gnu.org/copyleft/gpl.html, when you are in charge for Linux based software development



Richard Matthew Stallman (aka. rms) lauched:

GPLv3
Free Software
Free as in Freedom
GCC
FSF
FREE SOFTWARE
FOUNDATION
Emacs

https://en.wikipedia.org/wiki/Richard_Stallman

What is ‘free’ in GPL

- ❑ Free is not only price in common understanding

- ❑ But also
  - o Freedom of use
  - o Freedom of distribution
  - o Freedom of source code modification

- ❑ You have a choice to charge for your Linux based software package

What is ‘open source’ in GPL

- ❑ Not necessary to open ‘source code’ on web site
  - o When you are using object code (modified by yourself) for your private use only, you do not need to open your source code
  - o When you provide object code (modified by yourself) to limited person, such as Renesas employee, you need to open source code to the person who received object code.
    Not necessary to open source code to other person.

What is ‘as is’ and ‘warranty’ in GPL

❑ Program is provided ‘as is’ based, without any kind of warranty.

- o No guarantee / commitment to achieve any kind of functionality / performance

❑ Entire risk is with you : the person who use Linux.

❑ Copyright holder nor any person (including the person who distributes Linux) never take any kind of responsibility.

- o Not necessary to fix bug / modify source code
- o Not necessary to investigate any problem
- o Not necessary to answer against any question
- o All re-action is based on voluntary

Real-time Operating System (Real-time multi-task kernel)

Real-time kernel vs. OS

Real-time kernel
(or Real-time Operating System)

- mainly used for 16/32 bit MCU, such as M16C, H8/H8S, SH-2(2A)
- compact, smaller object size
- relatively fast
- No MMU support
- simple configuration
- static link

OS

- mainly used for 32 bit MPU, such as SH-4(4A)
- rich function support, larger object size
- relatively slow
- MMU support
- rich software sub-component support
- dynamic link

Note: above classification is example for study purpose, each OS/kernel has different function/specification/feature.

Real-time performance and multi-task control

❑ Real-time performance?

o Follow the constantly changing world we live in and control it with computer

Without real-time performance

- ➢ Need to wait for ready to use for a dozen of seconds after pressing start-up.
- ➢ User has to wait for getting the process done.

With real-time performance

- ➢ User can use immediately after pressing power-on.
- ➢ Video recording starts just from the moment user press the Record button.
- ➢ The motion and voice must be synchronized together.

❑ Multi-task?

o Executing task in parallel (or feel like parallel).

INDEPENDENCE Parallel operation (e.g.: in PC)

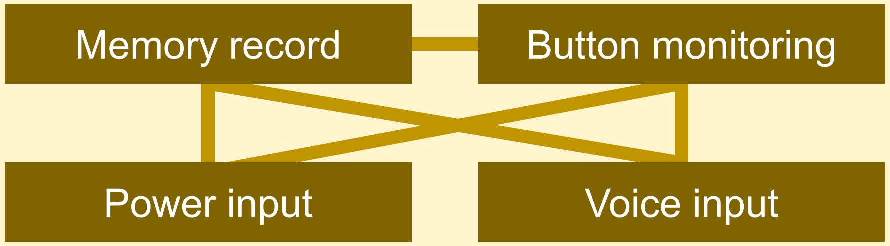
Spreadsheet

Mailer

Browser

Music player

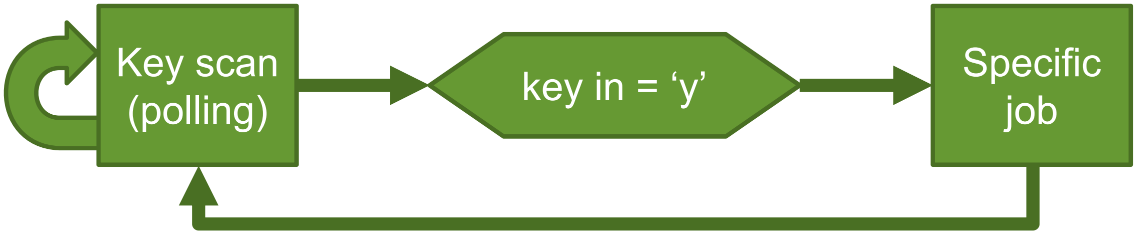
SYNCHRONOUS Parallel operation (e.g.: in DVC)



Real time performance

Real-time performance depends on latency against interrupts

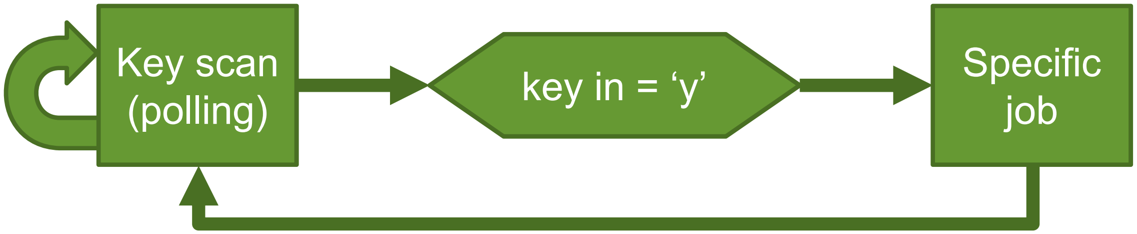
Old system (without interrupts) :



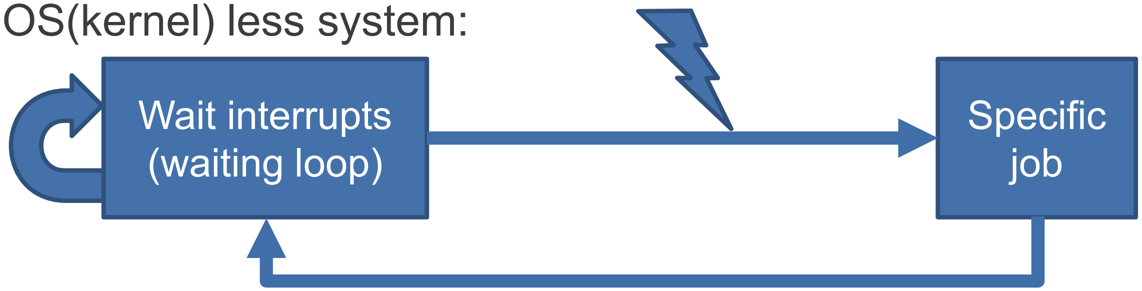
Real time performance

Real-time performance depends on latency against interrupts

Old system (without interrupts) :



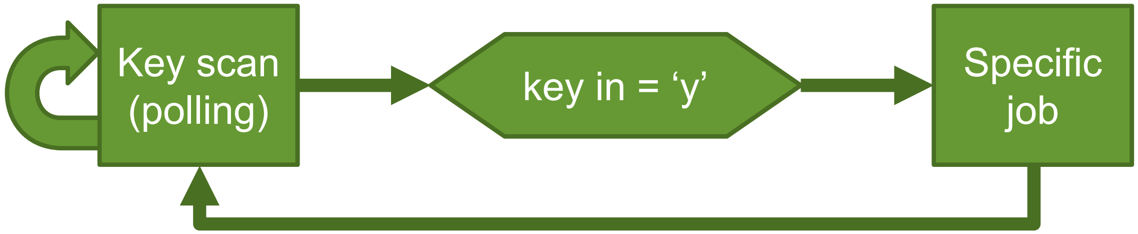
OS(kernel) less system:



Real time performance

Real-time performance depends on latency against interrupts

Old system (without interrupts) :

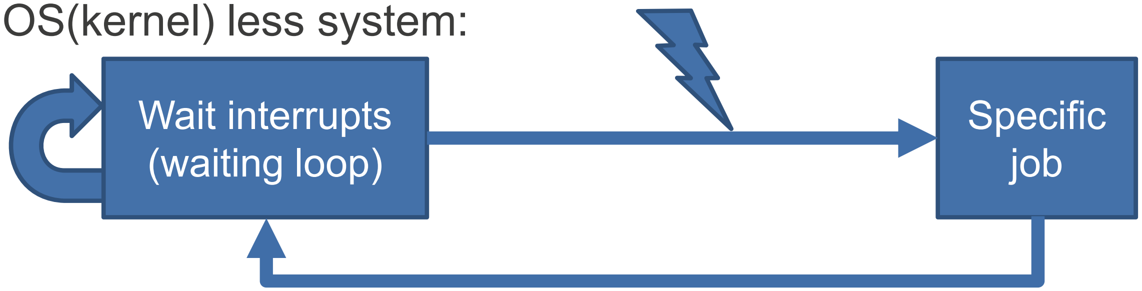


Key scan (polling)

key in = ‘y’

Specific job

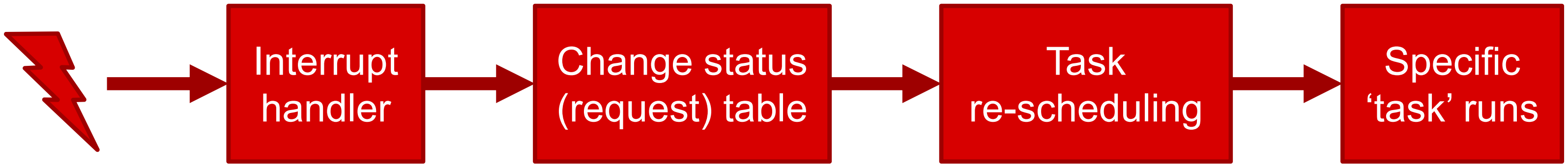
OS(kernel) less system:



Wait interrupts (waiting loop)

Specific job

OS (kernel) based system :



Interrupt handler

Change status (request) table

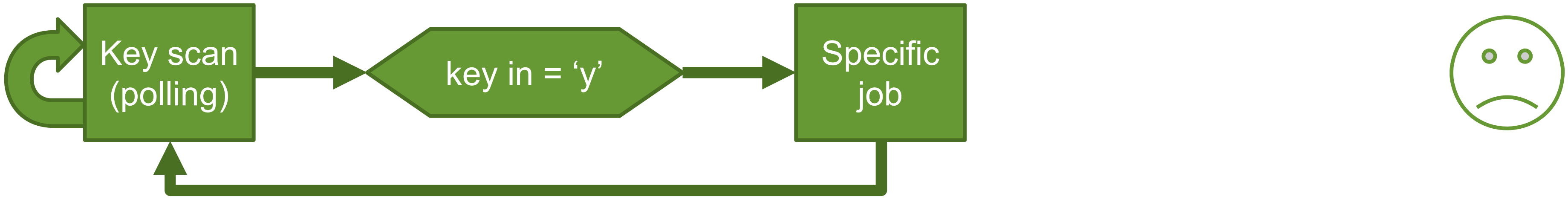
Task re-scheduling

Specific ‘task’ runs

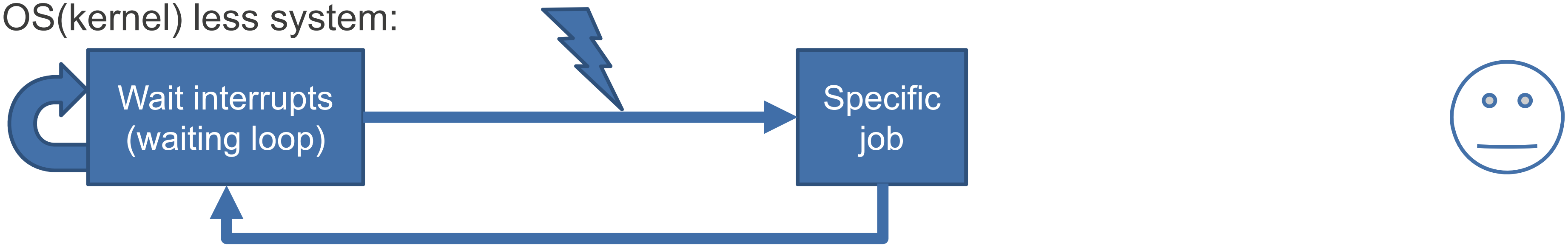
Real time performance

Real-time performance depends on latency against interrupts

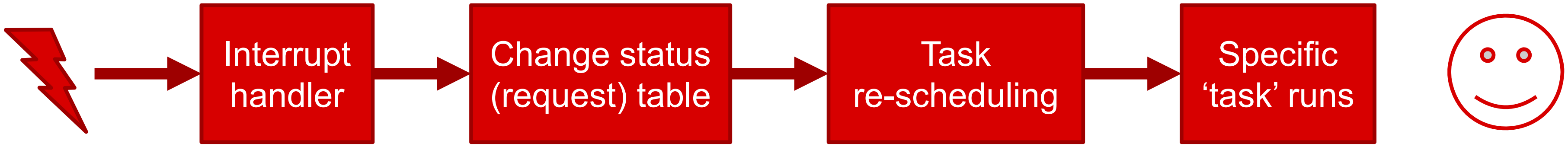
Old system (without interrupts) :



OS(kernel) less system:



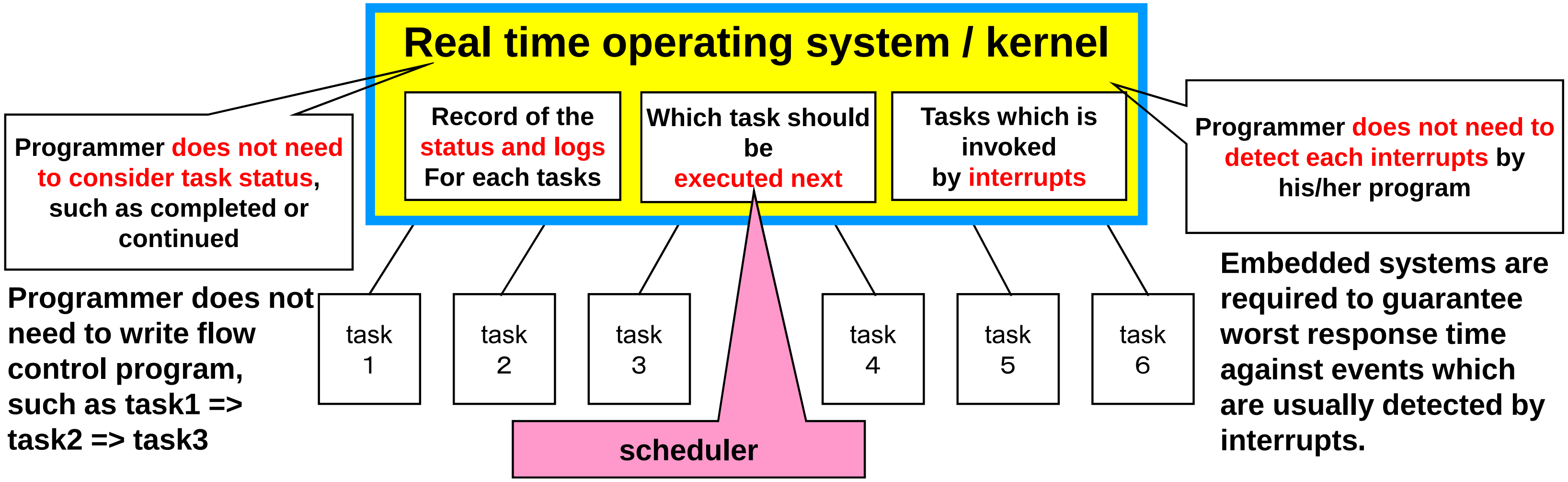
OS (kernel) based system :



Parallel processing under embedded real-time OS, kernel

❑ Need to register software groups (named ‘task’ which can be executed independently) to OS table

❑ Events (key input, timer overflow, etc.) generates interrupts and invokes related tasks



Real time operating system / kernel
Record of the status and logs For each tasks
Which task should be executed next
Tasks which is invoked by interrupts
task 1
task 2
task 3
task 4
task 5
task 6
scheduler

Programmer does not need to consider task status, such as completed or continued

Programmer does not need to detect each interrupts by his/her program

Programmer does not need to write flow control program, such as task1 => task2 => task3

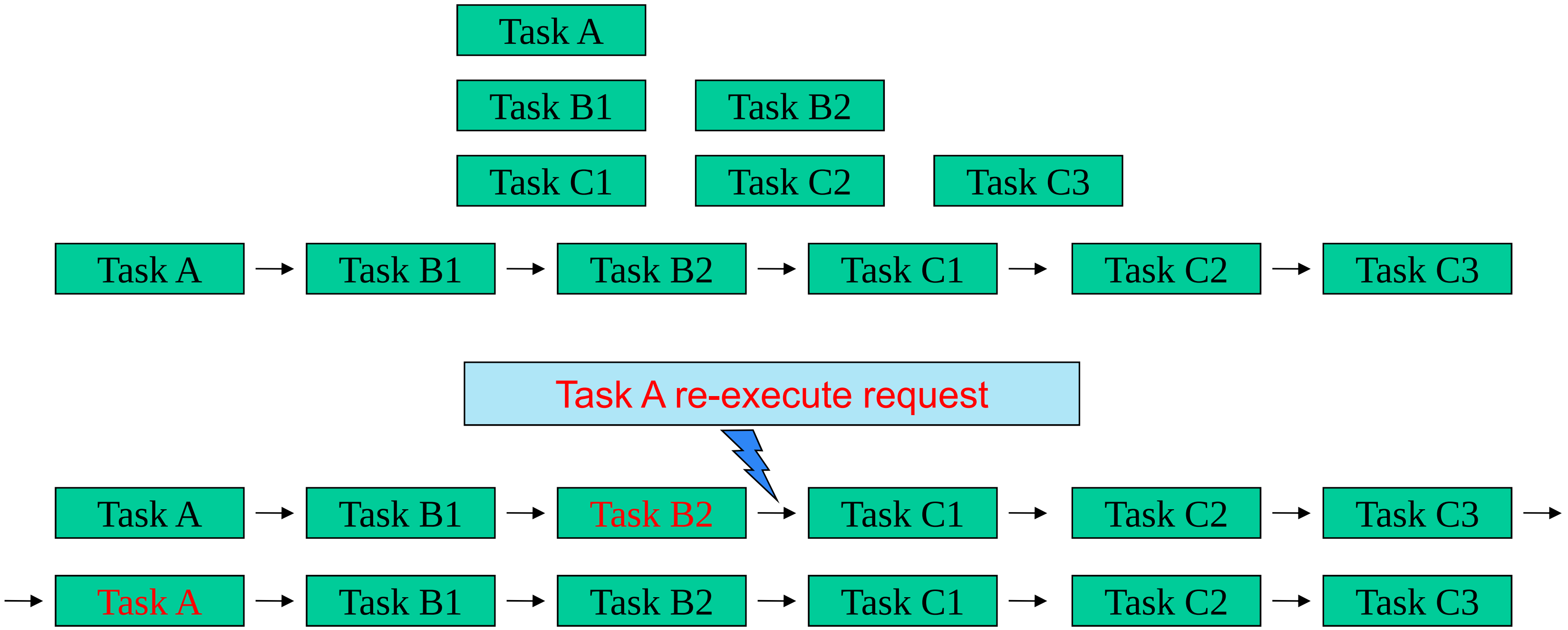
Embedded systems are required to guarantee worst response time against events which are usually detected by interrupts.

Application program is divided into several small groups which is called ‘task’.
Operating system provides task control function, i.e. CPU power is time sliced and in each small time slice, task is executed under pre-defined order.
There are two major algorithm for time slice control:

- Time shearing system is popular in PC application.
- Priority control system is popular in embedded application.

Scheduling: Priority and Round-robin

Round-robin scheduling : Common algorithm in computer system (or TSS: Time Sharing System)



Task A

Task B1 Task B2

Task C1 Task C2 Task C3

Task A → Task B1 → Task B2 → Task C1 → Task C2 → Task C3

Task A re-execute request

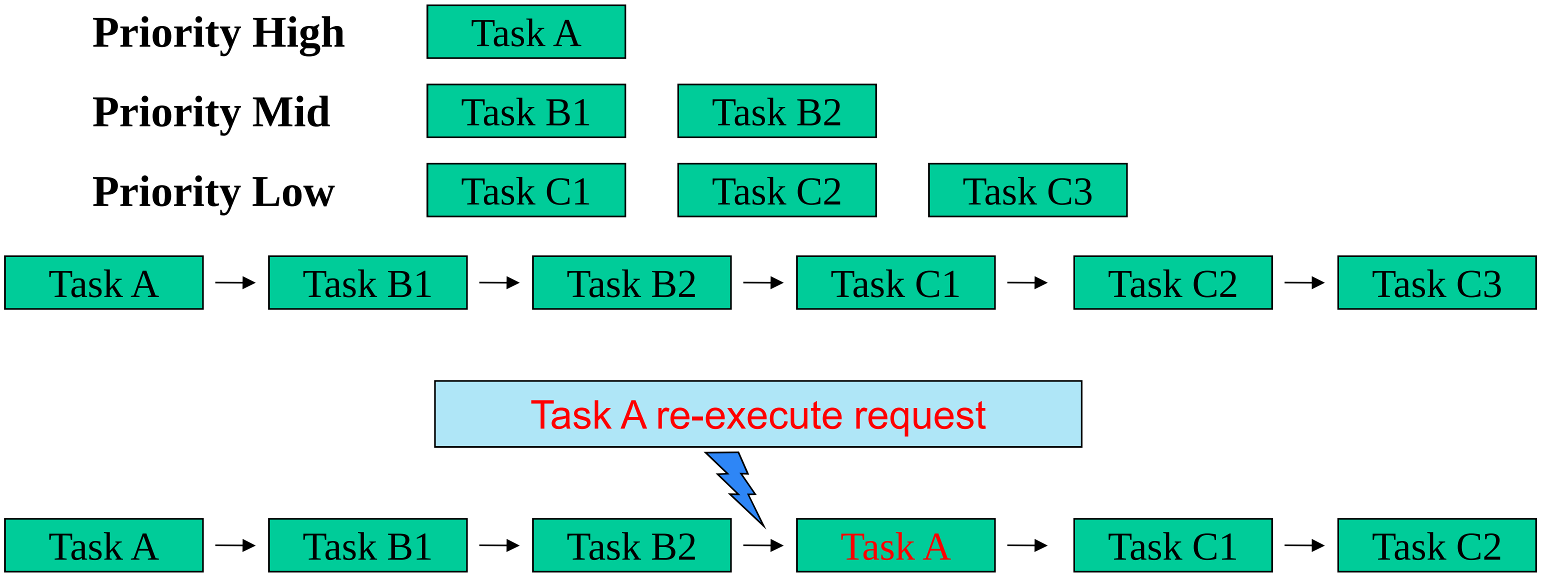
Task A → Task B1 → Task B2 → Task C1 → Task C2 → Task C3 →

→ Task A → Task B1 → Task B2 → Task C1 → Task C2 → Task C3

Every task will be executed in long term time period, but response time (from request to complete) is longer than priority control.

Scheduling: Priority and Round-robin

Priority scheduling : Major algorithm in embedded system



Priority High Task A

Priority Mid Task B1 Task B2

Priority Low Task C1 Task C2 Task C3

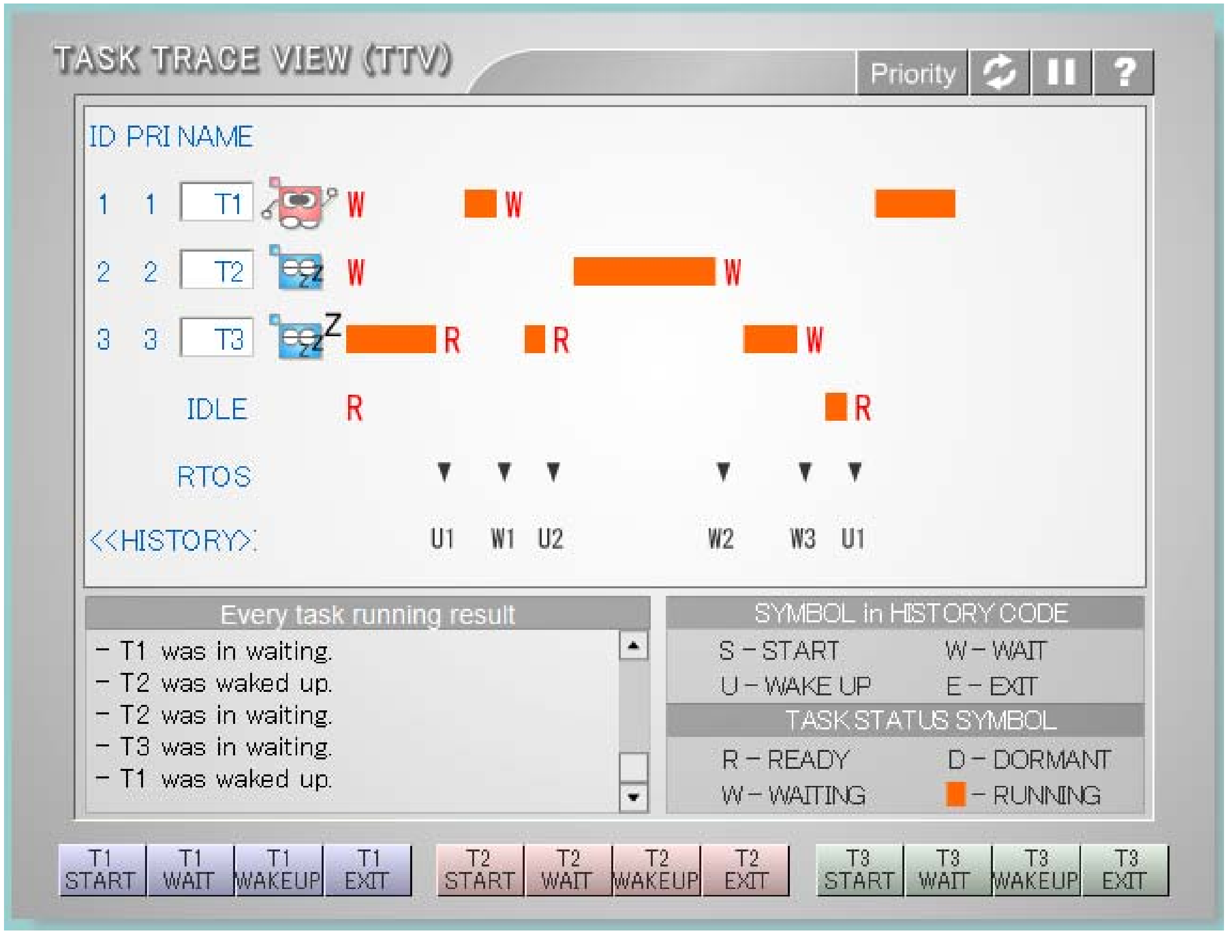
Task A → Task B1 → Task B2 → Task C1 → Task C2 → Task C3

Task A re-execute request

Task A → Task B1 → Task B2 → Task A → Task C1 → Task C2

Response time (from request to complete) is shorter than round-robin, but lower priority task may not executed forever if higher priority task runs frequently.

Scheduling: Visual Demo



Every task running result

- T1 was in waiting.
- T2 was waked up.
- T2 was in waiting.
- T3 was in waiting.
- T1 was waked up.

SYMBOL in HISTORY CODE

S – START	W – WAIT
U – WAKE UP	E – EXIT

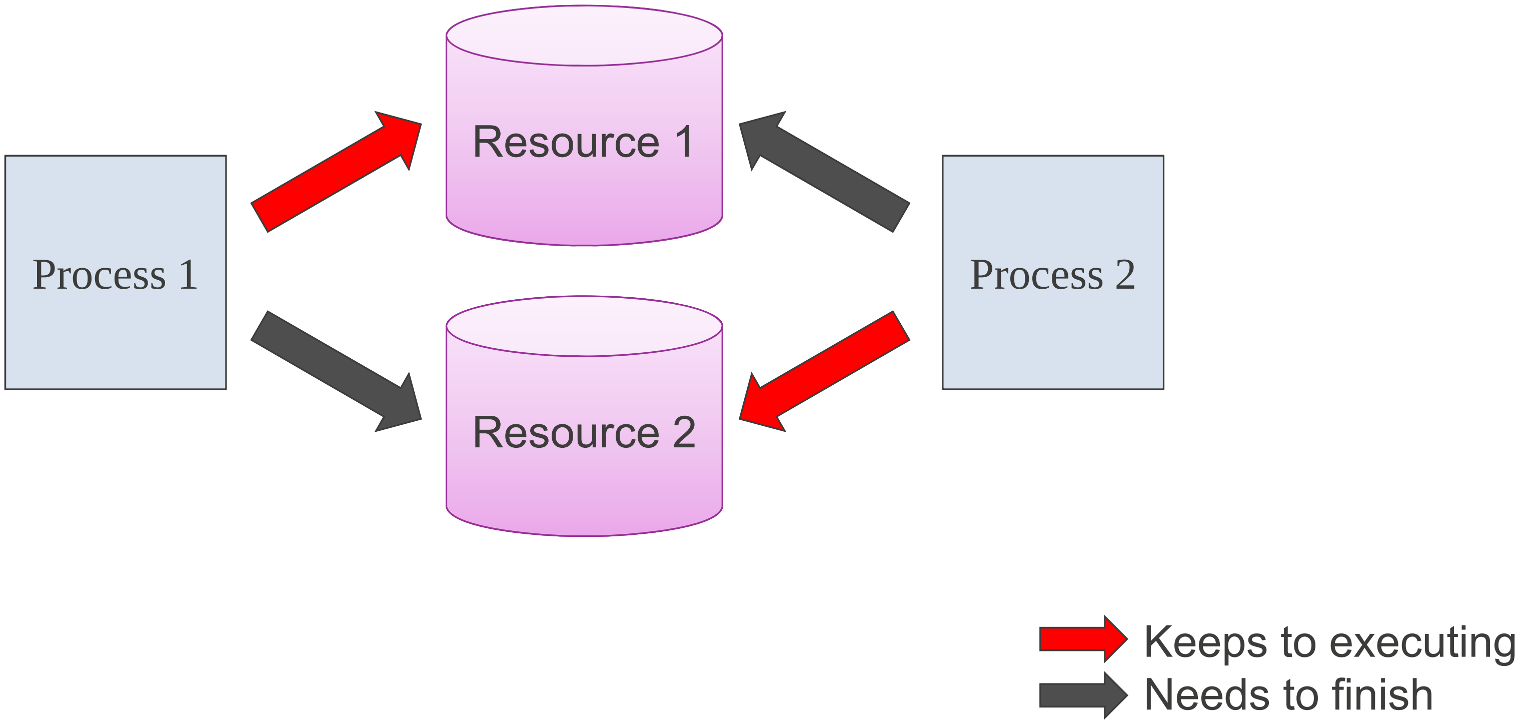
TASK STATUS SYMBOL

R – READY	D – DORMANT
W – WAITING	🟧 – RUNNING

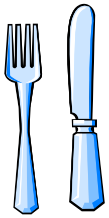
Resource control

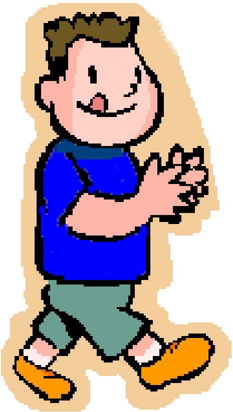
Deadlock

“If more than one resources are accessed from different process (task), there is always risk for deadlock”



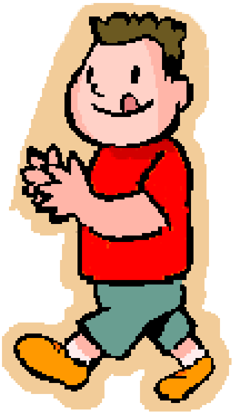
Resource control – Deadlock





Mr. A

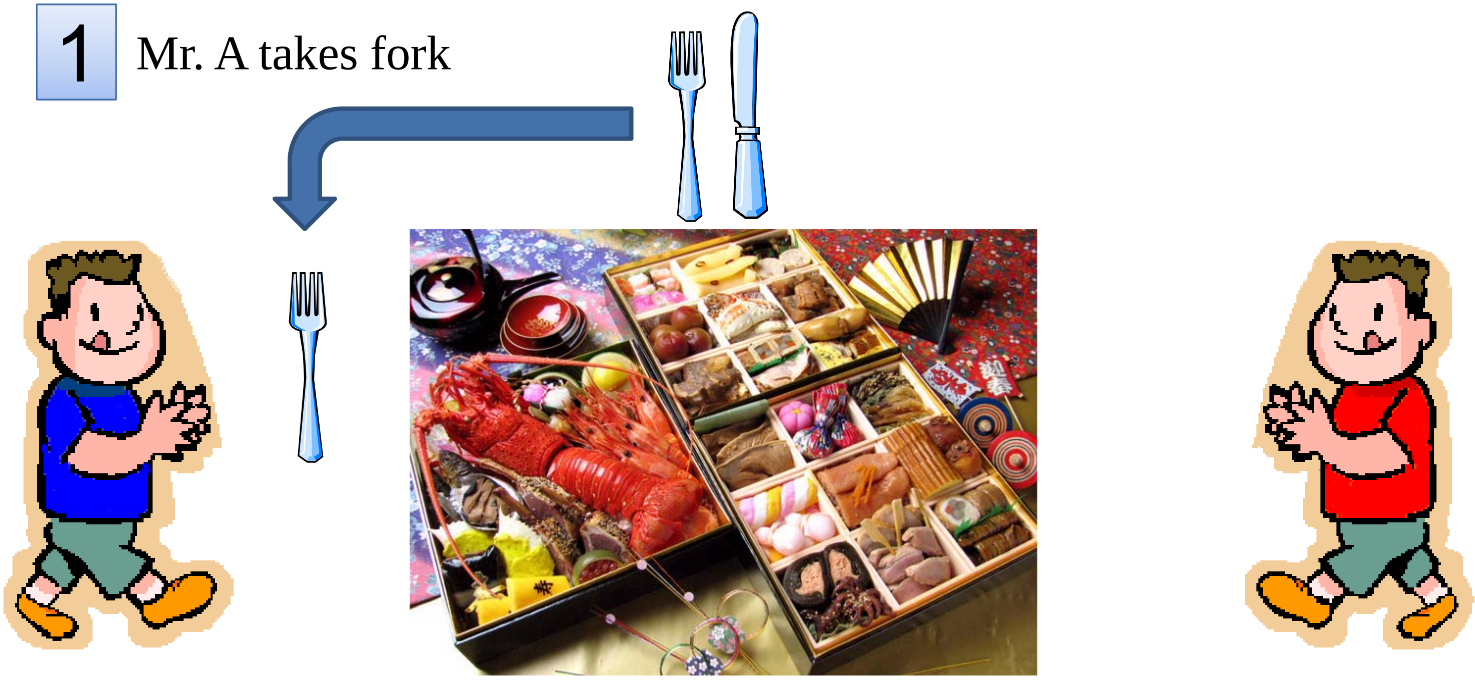
Osechi – Japanese new year foods
(http://en.wikipedia.org)



Mr. B

E.g. 2 (two) persons try to eat dinner, with sharing one fork and one knife.

Resource control – Deadlock



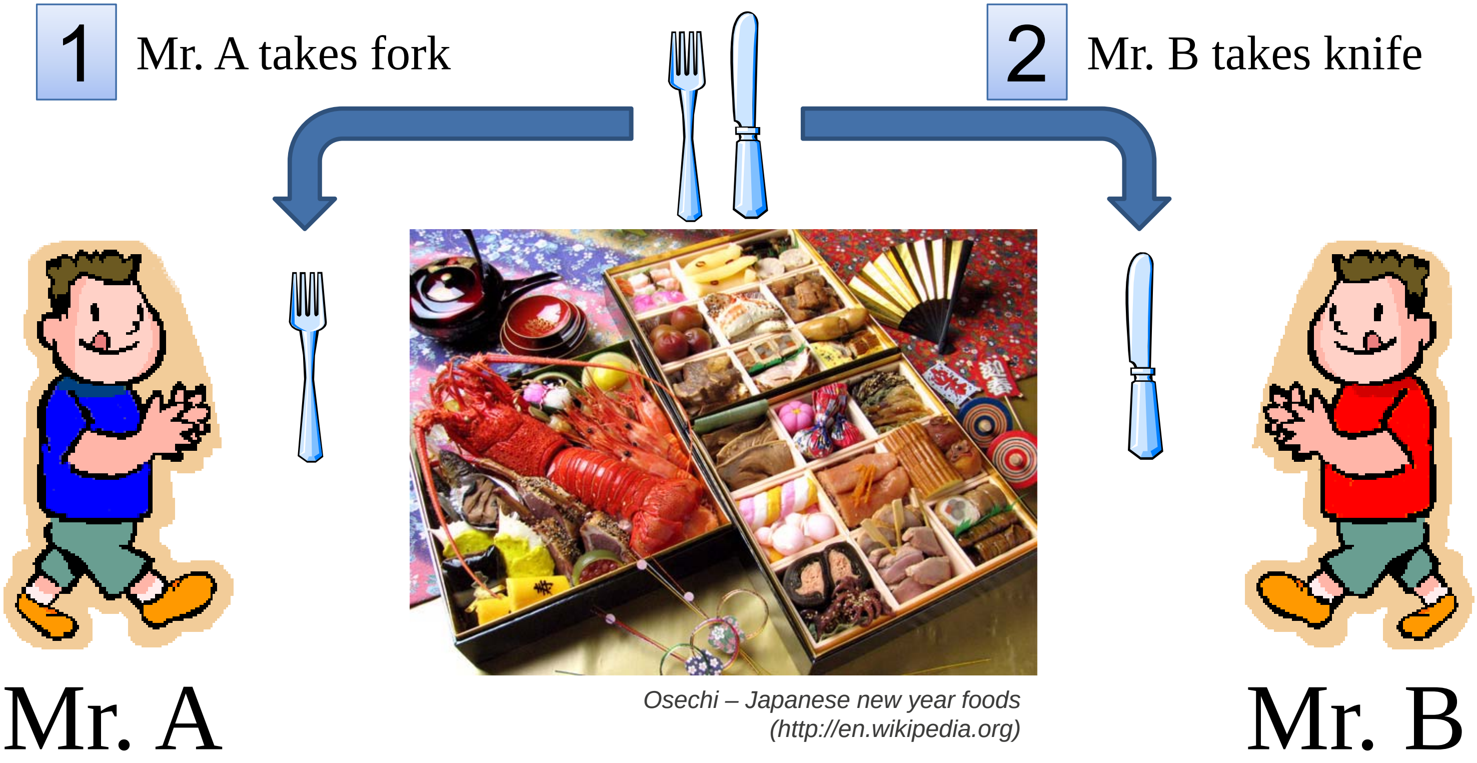
1 Mr. A takes fork

Mr. A

Osechi – Japanese new year foods (http://en.wikipedia.org)

Mr. B

Resource control – Deadlock



1 Mr. A takes fork

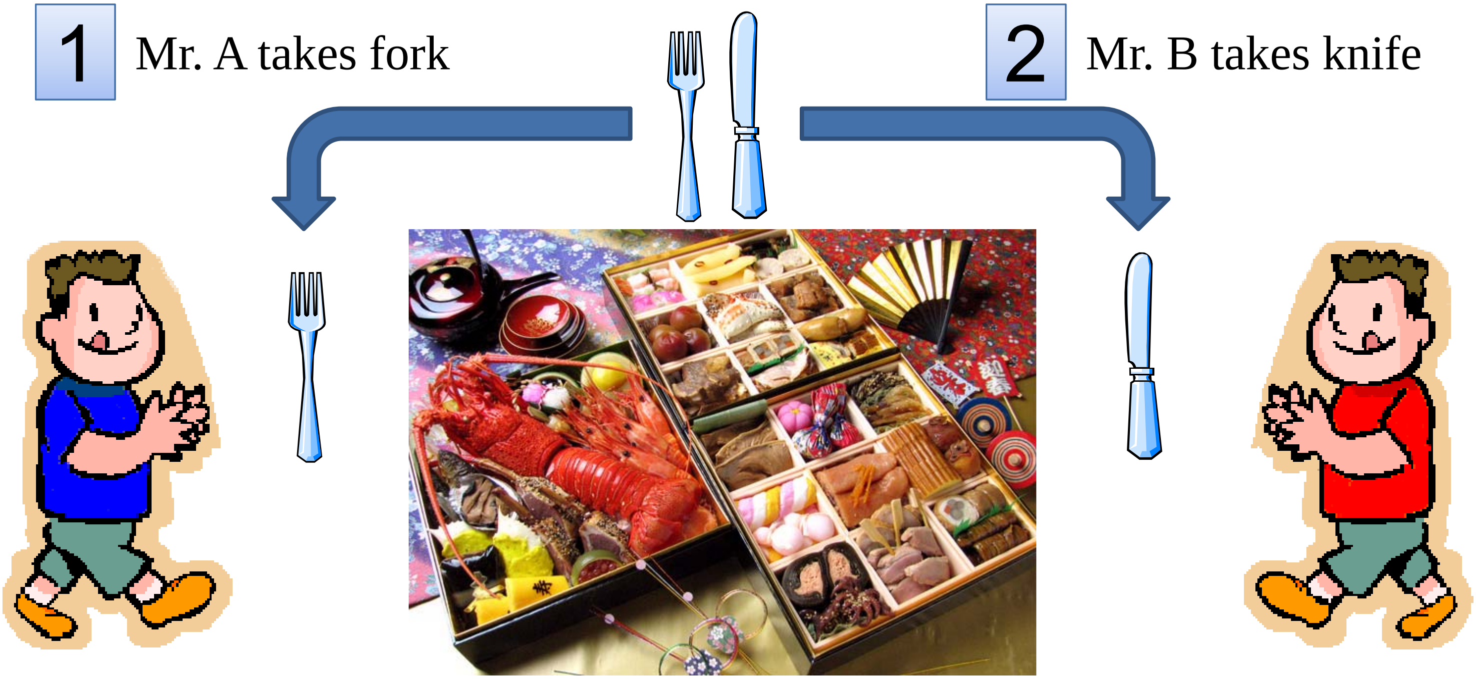
2 Mr. B takes knife

Mr. A

Mr. B

Osechi – Japanese new year foods (http://en.wikipedia.org)

Resource control – Deadlock



1 Mr. A takes fork

2 Mr. B takes knife

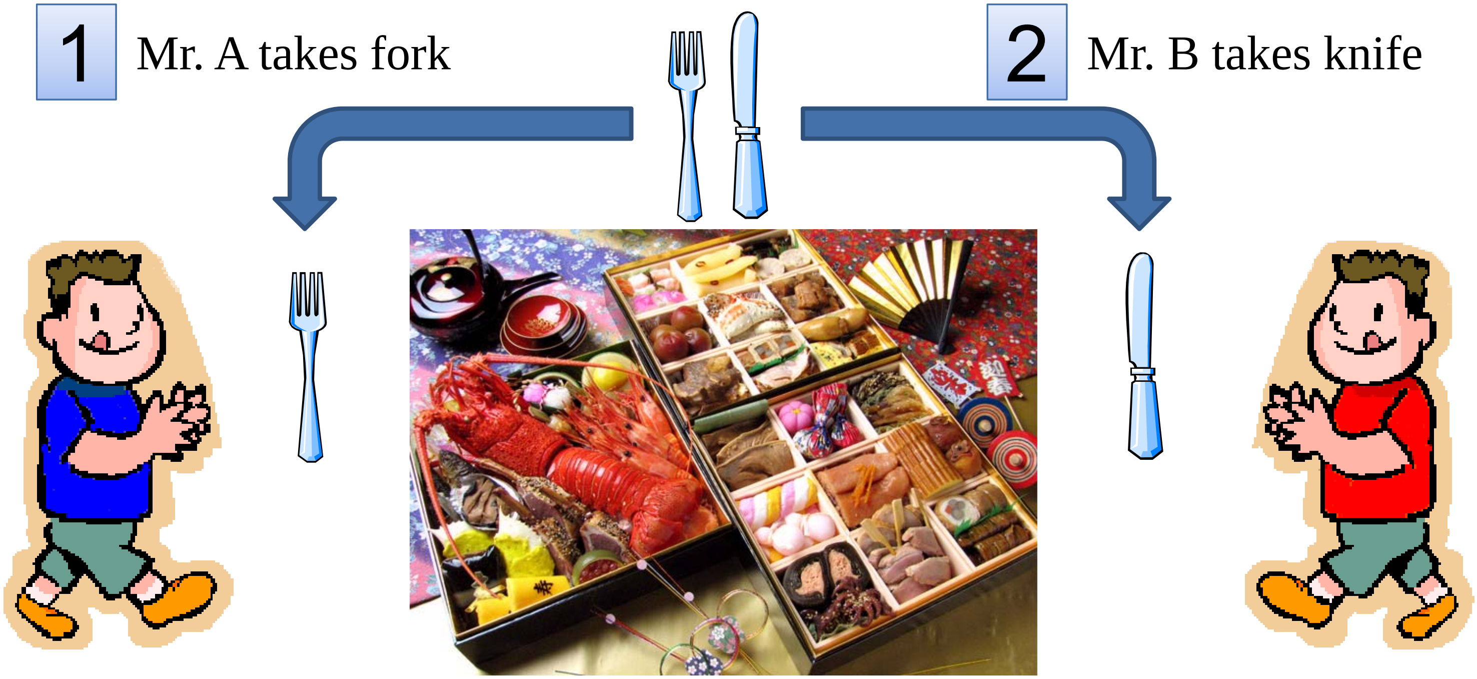
Mr. A

Mr. B

Osechi – Japanese new year foods (http://en.wikipedia.org)

3 Mr. A need fork, but fork is not available!
Mr. B need knife, but knife is not available!

Resource control – Deadlock



1 Mr. A takes fork

2 Mr. B takes knife

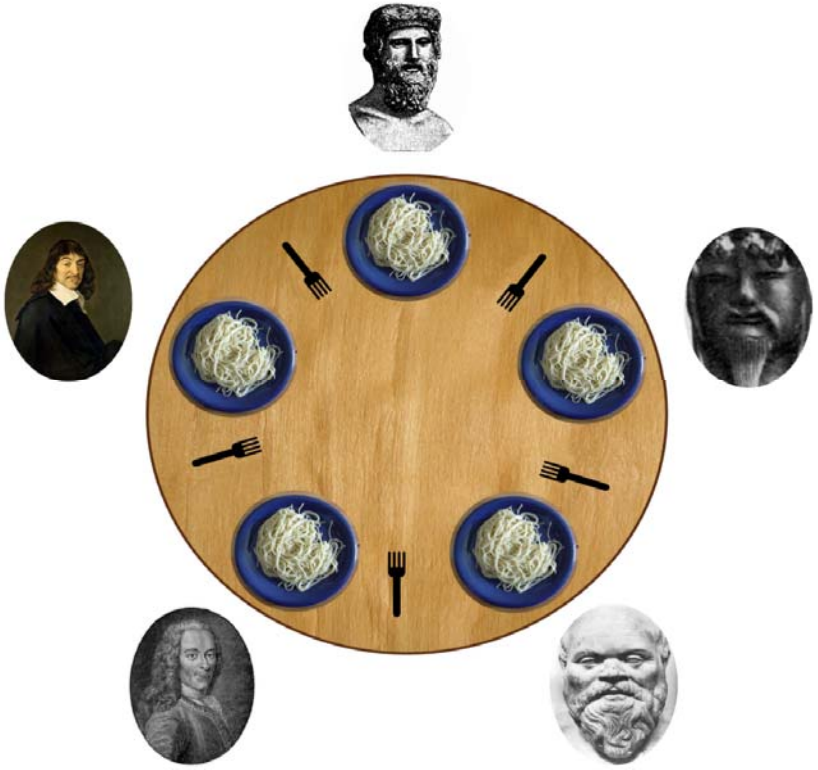
Mr. A

Mr. B

Osechi – Japanese new year foods
(http://en.wikipedia.org)

3 Mr. A need fork, but fork is not available!
Mr. B need knife, but knife is not available!
No one can eat!

Resource control – Deadlock



An illustration of the dining philosophers problem.
Philosophers: Plato, Confucius, Socrates, Voltaire and Descartes

(https://en.wikipedia.org/wiki/Dining_philosophers_problem)

Dining philosophers problem

The problem was designed to illustrate the challenges of avoiding deadlock, a system state in which no progress is possible. To see that a proper solution to this problem is not obvious, consider a proposal in which each philosopher is instructed to behave as follows:

- think until the left fork is available; when it is, pick it up;
- think until the right fork is available; when it is, pick it up;
- when both forks are held, eat for a fixed amount of time;
- then, put the right fork down;
- then, put the left fork down;
- repeat from the beginning.

Resource control – Deadlock

❑ Condition of deadlock ("Coffman conditions")

- **Mutual Exclusion**: At least one resource must be held in a non-shareable mode. Only one process can use the resource at any given instant of time.
- **Hold and Wait or Resource Holding**: A process is currently holding at least one resource and requesting additional resources which are being held by other processes.
- **No Preemption**: A resource can be released only voluntarily by the process holding it.
- **Circular Wait**: Processes must be waiting for resources which is being held by others in circle. In general, there is a set of waiting processes, P = {P1, P2, ..., Pn}, such that P1 is waiting for a resource held by P2, P2 is waiting for a resource held by P3 and so on until Pn is waiting for a resource held by P1.

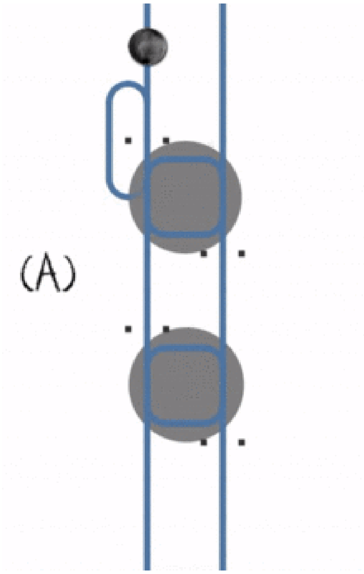
Resource control – Deadlock

❑ Condition of deadlock (“Coffman conditions”)

- o **Mutual Exclusion**: At least one resource must be held in a non-shareable mode. Only one process can use the resource at any given instant of time.
- o **Hold and Wait or Resource Holding**: A process is currently holding at least one resource and requesting additional resources which are being held by other processes.
- o **No Preemption**: A resource can be released only voluntarily by the process holding it.
- o **Circular Wait**: Processes must be waiting for resources which is being held by others in circle. In general, there is a set of waiting processes, P = {P1, P2, ..., Pn}, such that P1 is waiting for a resource held by P2, P2 is waiting for a resource held by P3 and so on until Pn is waiting for a resource held by P1.

Prevent one of the four Coffman conditions from occurring ➔ Prevent Deadlock.

Resource control – Deadlock



Two processes competing for two resources in opposite order.

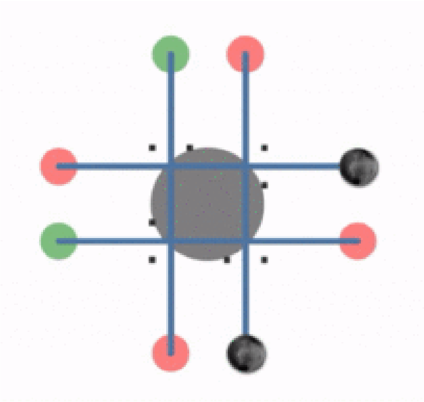
a) 1 single process goes through.

b) The later process has to wait.

c) A deadlock occurs when the first process locks the first resource at the same time as the second process locks the second resource.

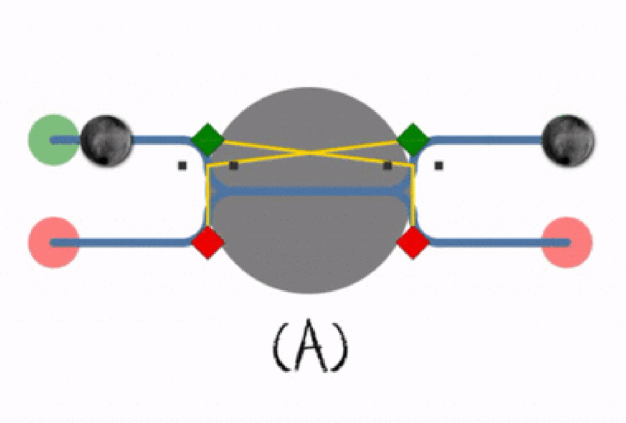
d) The deadlock can be resolved by cancelling and restarting the first process.

Resource control – Deadlock



Four processes (blue lines) compete for one resource (grey circle), following a right-before-left policy. A deadlock occurs when all processes lock the resource simultaneously (black lines). **The deadlock can be resolved by breaking the symmetry.**

Resource control – Deadlock



a) Two processes competing for one resource, following a first-come, first-served policy.
b) A deadlock occurs when both processes lock the resource simultaneously.
c) The deadlock can be resolved by breaking the symmetry of the locks.
d) The deadlock can be prevented by breaking the symmetry of the locking mechanism.

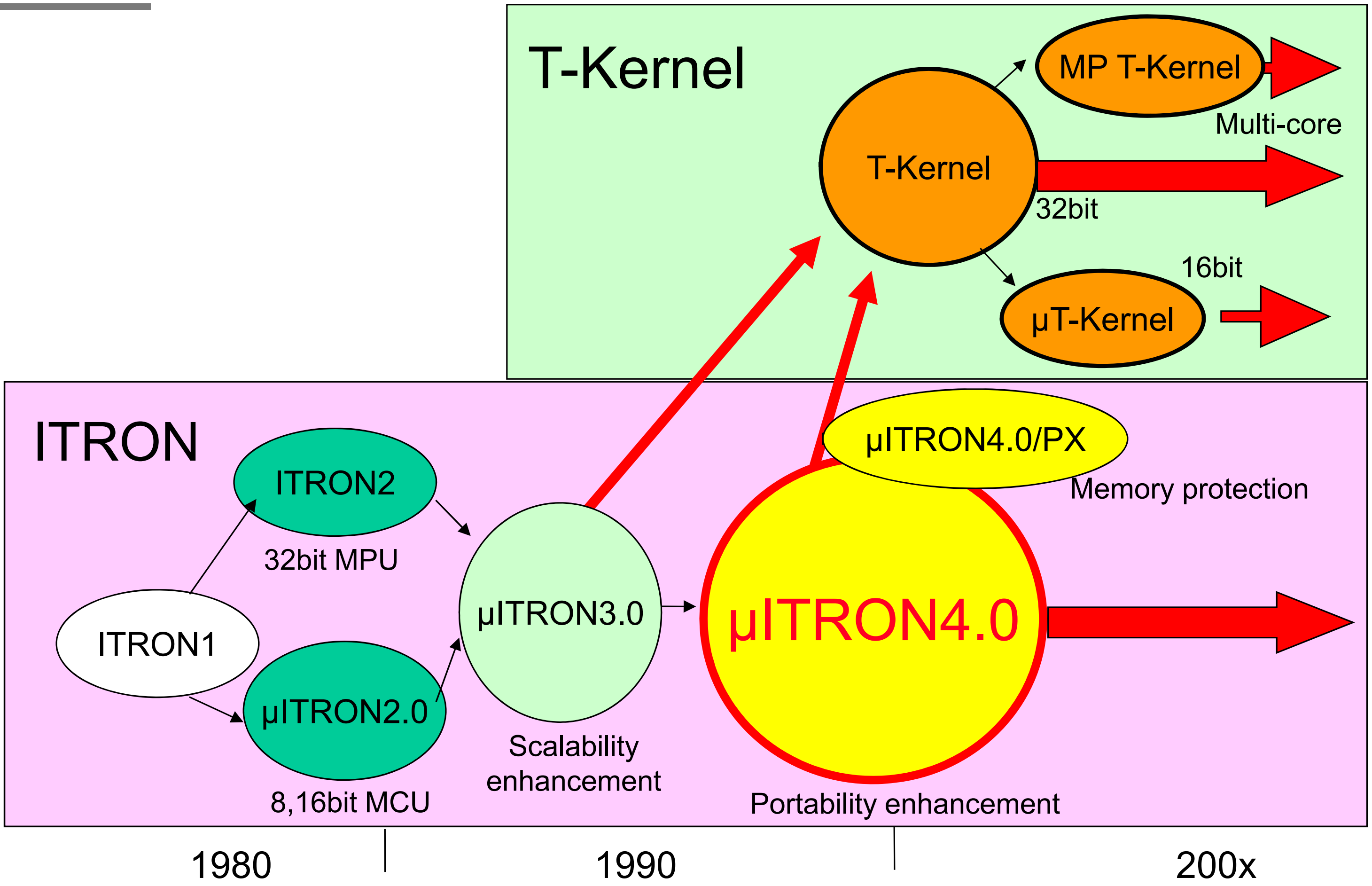
uITRON outline

❑ ITRON is a Japanese open standard for RTOS initiated in 1984 under guidance of Ken Sakamura, a University of Tokyo Professor. This project aims to standardize the RTOS and related specifications for embedded systems, particularly small-scale embedded systems. The ITRON RTOS is targeted for consumer electronic devices, such as mobile phones and fax machines. Various vendors sell their own implementations of the RTOS.

❑ μITRON (spelled uITRON or microITRON) are the name of RTOS specifications coming out of ITRON projects. 'μ' means that the particular specification is meant for the smaller 8-bit or 16-bit CPU targets. Specifications are available for free. Commercial implementations are available, and offered under many different licenses.

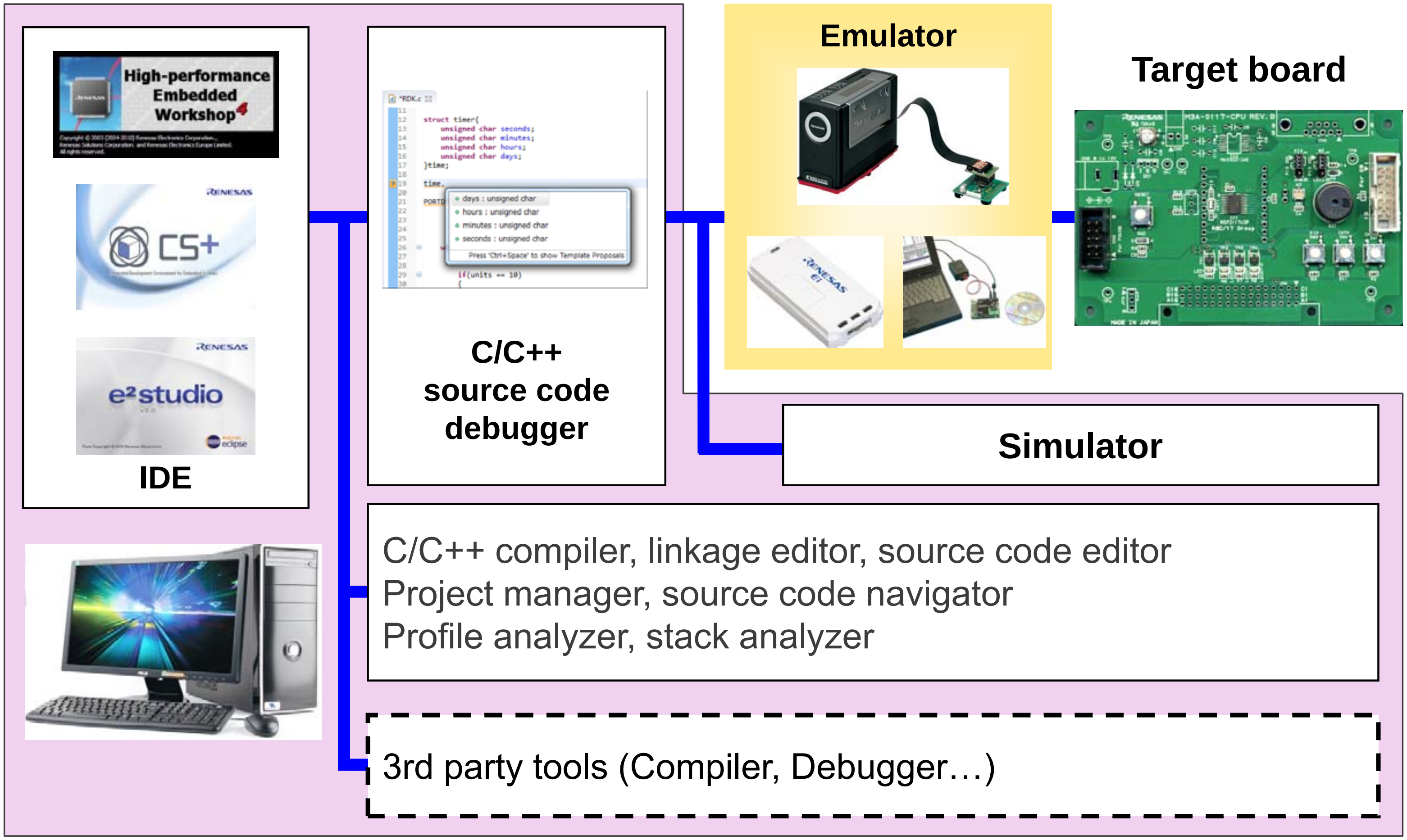
❑ Renesas Electronics Corp. has been participated in the TRON project up to today since the project was established. [HI series OS] is a real-time operating system for the Renesas H8/SH microcomputers which is compliant with uITRON specifications. Refer to the TRON Association website for detailed information on the ITRON specification and the μITRON specification.

uITRON outline – Roadmap



Integrate development environment

Development tools system configuration



*RDK.c

```
struct timer{
    unsigned char seconds;
    unsigned char minutes;
    unsigned char hours;
    unsigned char days;
}time;

time.

PORTD
```

- days : unsigned char
- hours : unsigned char
- minutes : unsigned char
- seconds : unsigned char

Press 'Ctrl+Space' to show Template Proposals

```
        if(units == 10)
        {
```

IDE

C/C++ source code debugger

Emulator

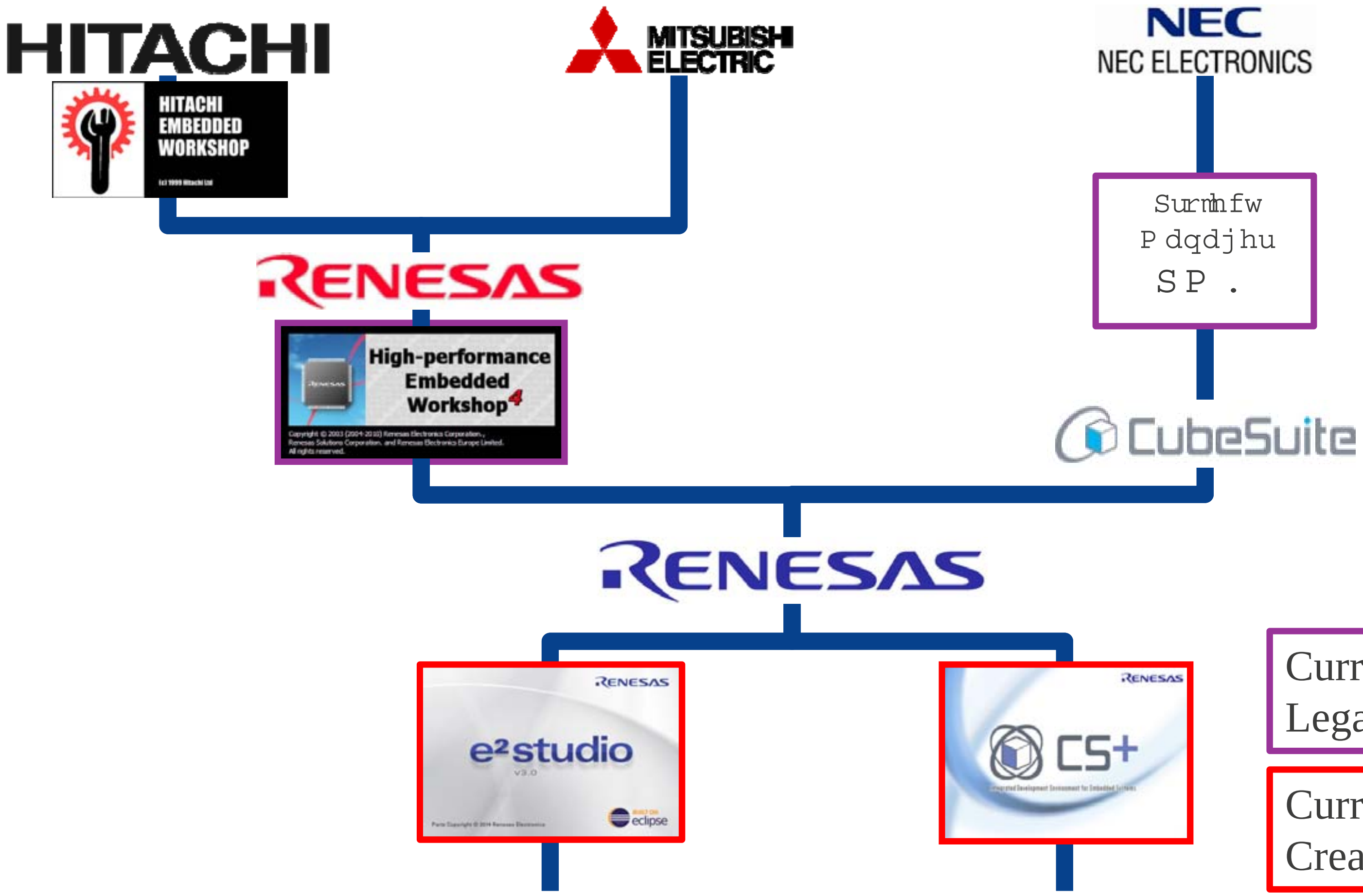
Target board

Simulator

C/C++ compiler, linkage editor, source code editor
Project manager, source code navigator
Profile analyzer, stack analyzer

3rd party tools (Compiler, Debugger…)

Renesas IDE (Integrated Development Environment)



Currently in use mainly for Legacy Projects

Currently in use for Newly Created Projects

Renesas IDE (Integrated Development Environment)

Development information

	Surmhfw Pdqdjhu SP .			
Latest version	V6.3x (2007)	V.4.09.01 (Jun, 2012)	V6.00.00 (Jul, 2017)	V6.1.0.0 (Oct, 2017)
Plan of next version	*none*	*none*	V6.01.00 (Jan, 2018)	V6.2.0.0 (Jan, 2018)
Copyright	Full	Full	Full	Part
Development status	*Maintenance*	*Maintenance*	In development	In development

Renesas IDE (Integrated Development Environment)

Supported devices (current version)

Surmhfw Pdqdjhu SP .	 High-performance Embedded Workshop4 Copyright © 2003 (2004-2010) Renesas Electronics Corporation., Renesas Solutions Corporation. and Renesas Electronics Europe Limited. All rights reserved.	 RENESAS CS+	 RENESAS e²studio v3.0 eclipse
V850 78K	H8 Tiny M16C Platform M32R SuperH RX 740 R8C	RL78 RX V850 RH850 78K R8C …	RL78 RX SuperH RH850 RZ RENESAS Synergy™ …

➔ Choose best device for product first. Then, selected suitable IDE.

Renesas IDE (Integrated Development Environment)

Supported Compilers

Surmhfw P dqdjhu SP .			
Renesas Compilers	Renesas Compilers	Renesas Compilers Green Hills	Renesas Compilers KPIT GNU IAR Green Hills

Supported OS

Surmhfw P dqdjhu SP .			
Windows® XP Windows® 2000	Windows® 7 Windows Vista® Windows® XP	Windows® 7, 8.x, 10 Windows Vista®	Windows® 7, 8.x, 10 Linux MacOS

Renesas IDE (Integrated Development Environment)

Supported Renesas Debugger Tools

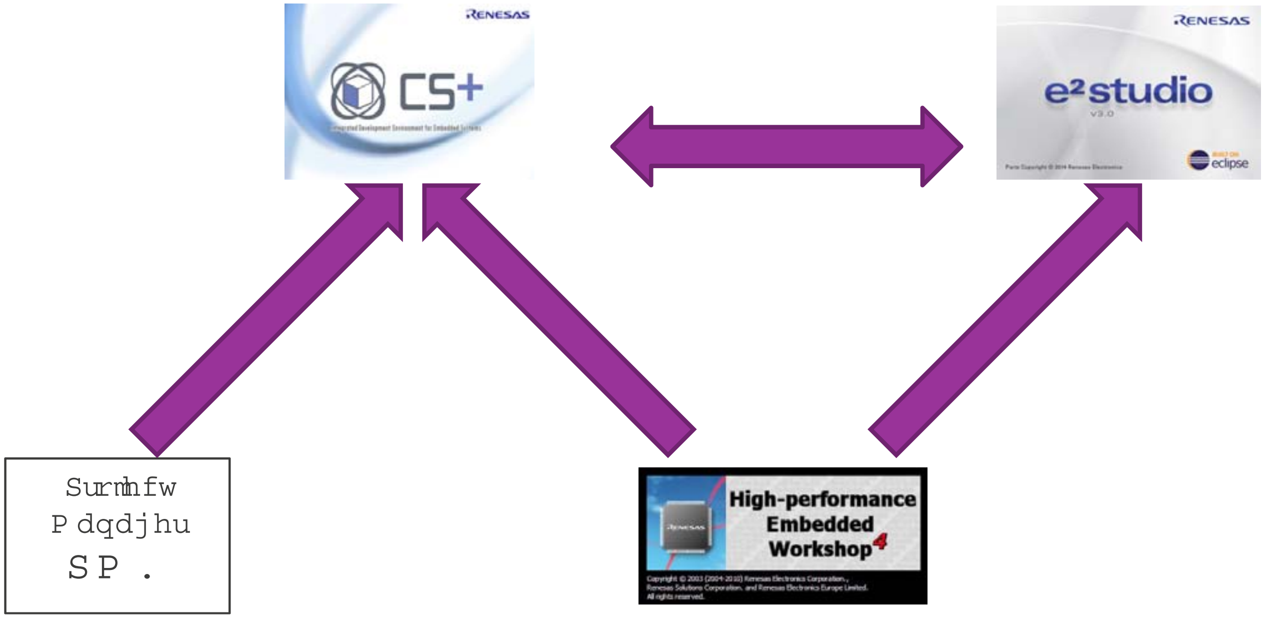
Surmhfw Pdqdjhu SP .			
IECUBE MINICUBE2	E200F E100 PC7501 PC4701 R0E521000CPE00 M38000T2-CPE E10A E1/E20 E10T E30A …	IECUBE IECUBE2 E1/E20 MINICUBE MINICUBE2	E1/E20 IECUBE E10A Segger J-Link

➔ Refer section 8 and 9 for more information about Renesas Debugger Tool

Renesas IDE (Integrated Development Environment)

Porting projects among integrated development environments

Projects can be ported between the several IDEs in following directions:

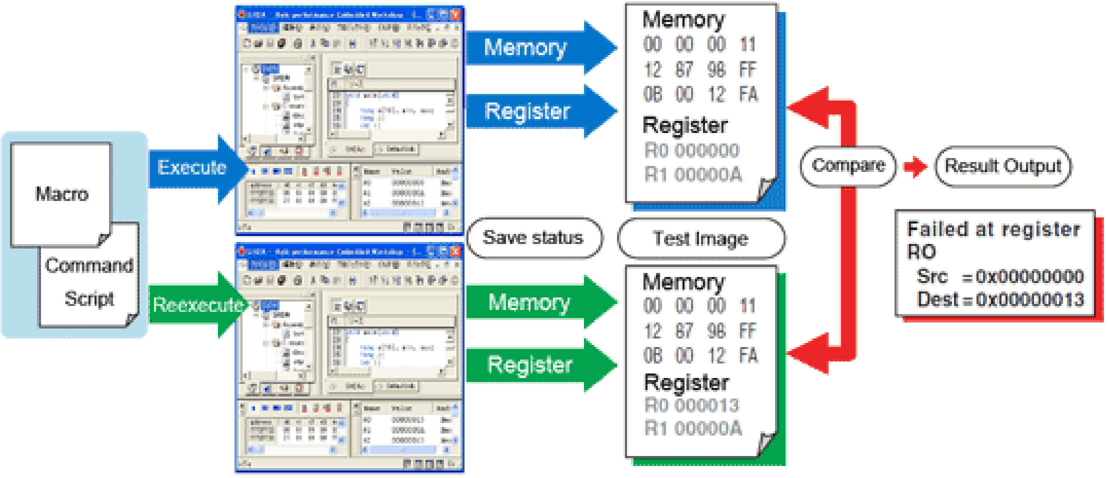


Renesas IDE (Integrated Development Environment)

Special feature – HEW

HEW has long development time, it is powerful and supports advanced debugging features such as code profiling, performance analysis, code coverage, trace features…

➔ We are selecting the most suitable features to implement for newer IDE (CubeSuite+, e^2 studio)



```
Memory
00  00  00  11
12  87  98  FF
0B  00  12  FA
Register
R0 000000
R1 00000A
```

```
Memory
00  00  00  11
12  87  98  FF
0B  00  12  FA
Register
R0 000013
R1 00000A
```

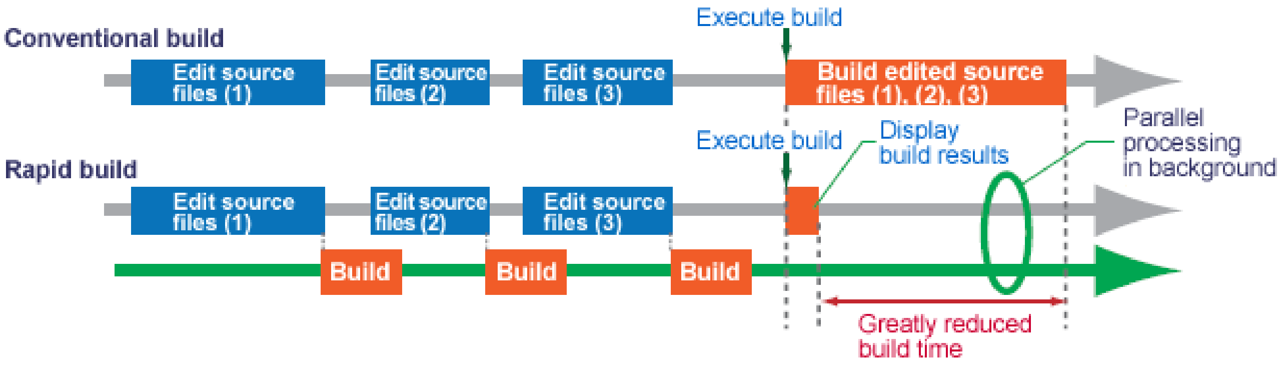
```
Failed at register
RO
 Src  =0x00000000
 Dest =0x00000013
```

The test support function, if combined with **the macro generation support function**, becomes even more convenient. For example, if you have a test procedure prepared in a macro file and the expected value of test prepared in a test image file available, a series of repeat tests comprised of test execution and comparison of test result can be performed efficiently.

Renesas IDE (Integrated Development Environment)

Special feature – CubeSuite+

CubeSuite+ can be used with the newly families as well as the conventional MCUs, RL78 family, RX family, V850 family, 78K0R, and 78K0. Renesas will continue to expand the number of development tools supported by CubeSuite+, persistently making your development environment better and faster.



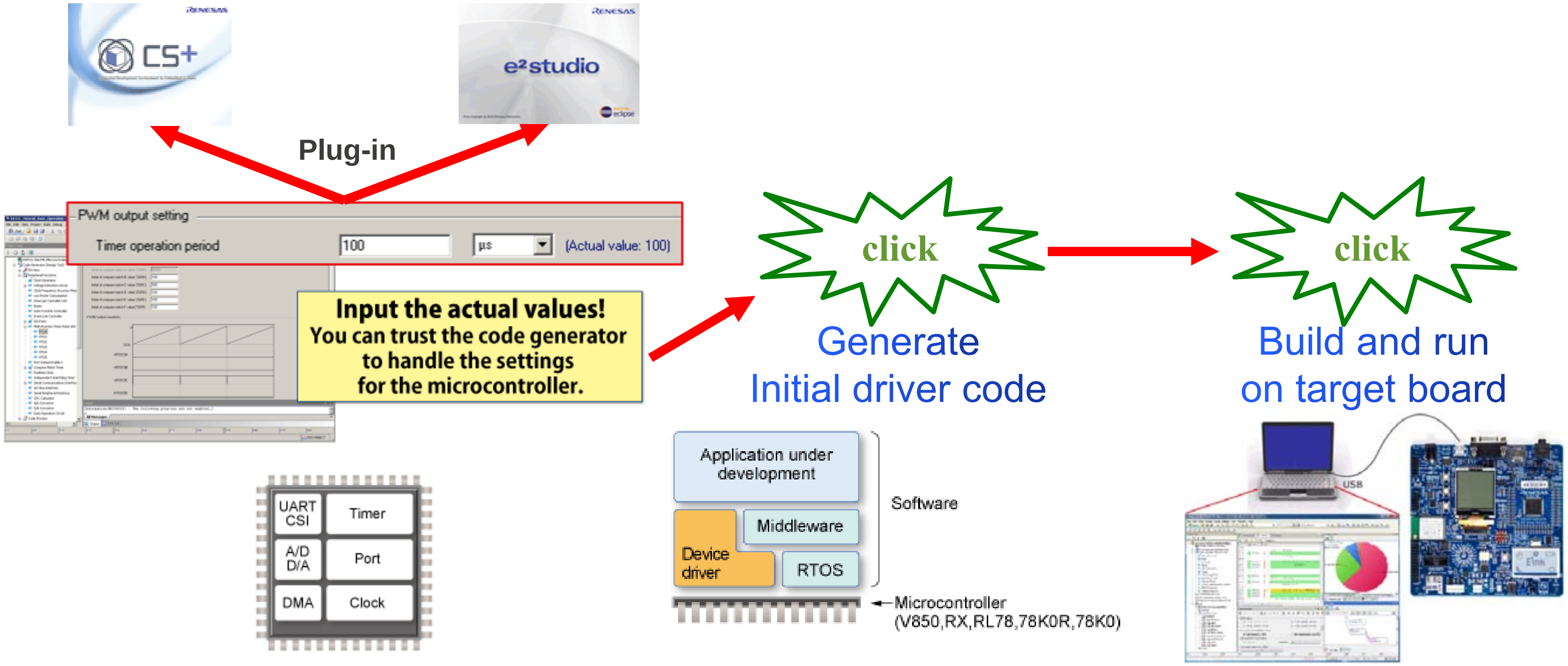
Rapid build enables greatly reduced build time

Conventional development environments require the user to edit all source files first, and then execute a build for the entire program, resulting in a lengthy build time.
CubeSuite+ features a "Rapid Build" function that automatically runs the build function in the background each time a source file is changed or saved, greatly reducing overall build time.

Renesas IDE (Integrated Development Environment)

Special feature – Code Generation (aka. CG)

Nowadays, developers do not have much time to refer HW Manual for coding (or they do not like to read).
GC supports developers by Simple GUI setting for creating Driver Code.
CG now is the most popular one among Renesas' tools.



PWM output setting

Timer operation period 100 µs (Actual value: 100)

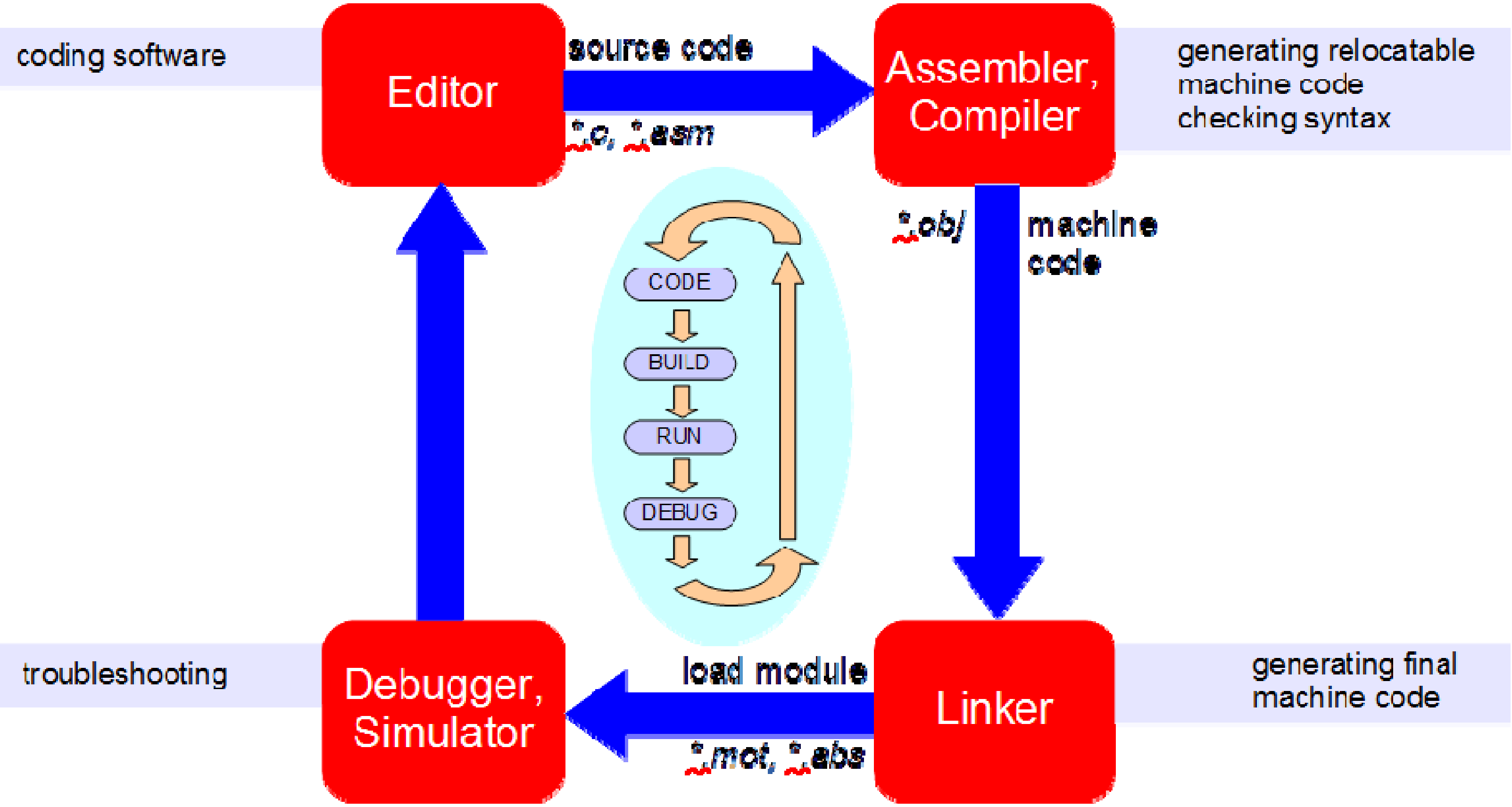
Input the actual values!
You can trust the code generator to handle the settings for the microcontroller.

Cross software

- C/C++ compiler
- Linkage editor
- MISRA-C rule checker

Cross software

- ❑ We need multiple-tools to build Load Module from Source code.



C/C++ compiler

- ❑ The Renesas C/C++ Compiler is an optimizing ANSI C and ANSI C++ compiler for the specific microprocessors.

- ❑ In addition to full ANSI C support, the compiler provides #pragma language extensions and command-line switches to support target specific features and extended compiler functionality.

- ❑ The C/C++ compiler has powerful and reliable code generation facilities for the targets. A variety of optimization features allow you to generate highly optimized PROM code. In particular, code can be optimized for size or speed to match the requirements of the particular application being developed.

Linkage editor

❑ Outline

- Linkage editor input the object programs from the compiler or assembler and output the load modules or library files.

❑ Optimization

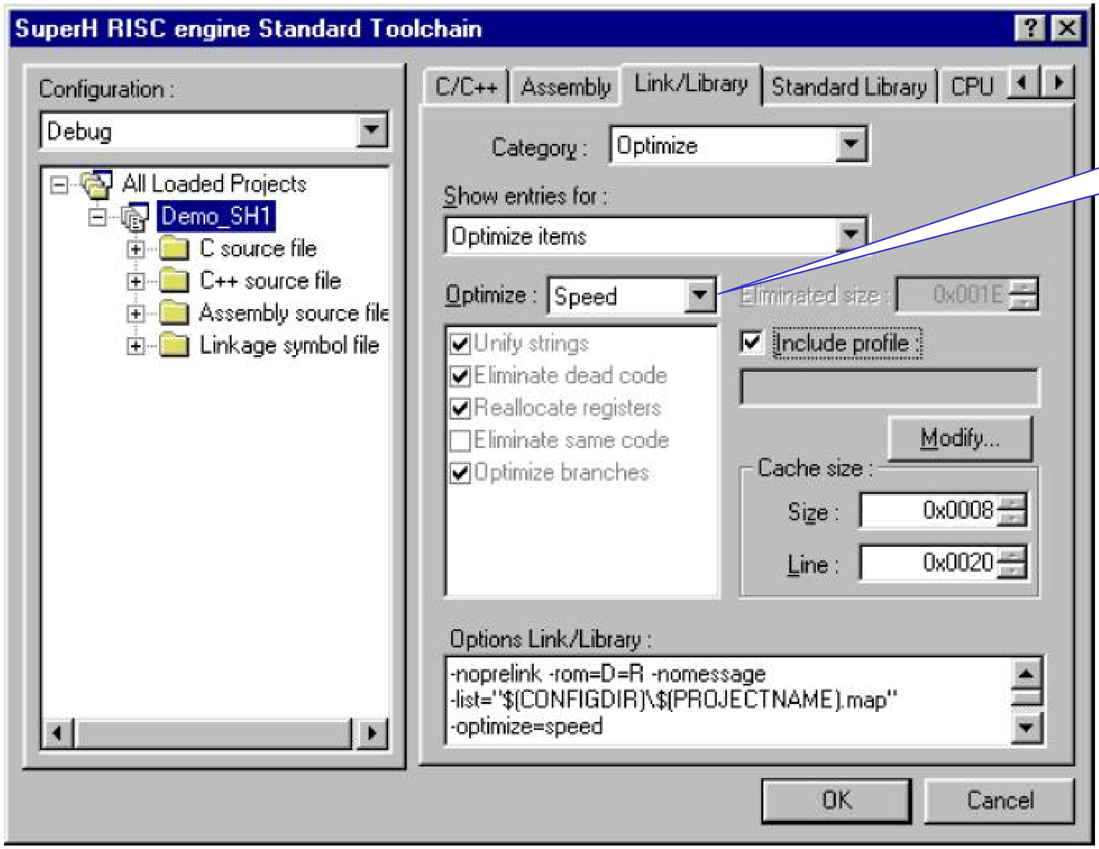
- Linkage editor analyzing the relationship between the object programs.
- MAP optimizing: Optimizing linker re-compile the program using symbol allocated address defined by compiling or linking.
- Global optimization: Combine all compiler objects to an unique object file.

❑ Output file

- Linkage editor output the file in the specified format: Absolute(ELF), Motorola S-type, HEX type, Binary type and symbol reference list.

Linkage editor

Selection of speed optimize or size optimization



Configuration : Debug

Category : Optimize

Show entries for : Optimize items

Optimize : Speed

Eliminated size : 0x001E

☑ Unify strings
☑ Eliminate dead code
☑ Reallocate registers
☐ Eliminate same code
☑ Optimize branches

☑ Include profile :

Cache size :
Size : 0x0008
Line : 0x0020

Options Link/Library :

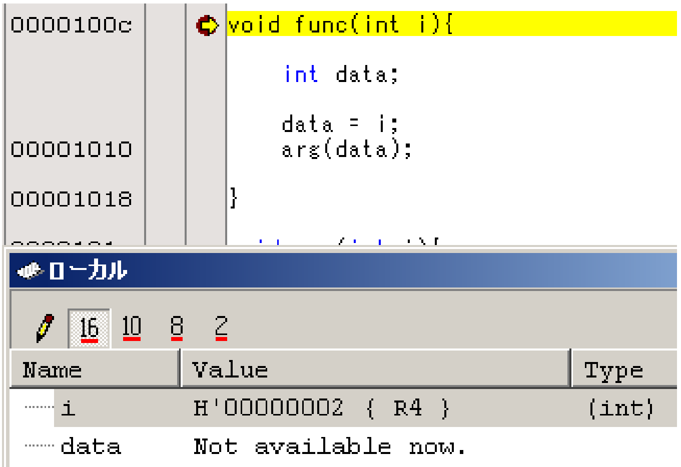
```
-noprelink -rom=D=R -nomessage
-list="$(CONFIGDIR)\$(PROJECTNAME).map"
-optimize=speed
```

Speed/size optimization

Linkage editor

Selection of debug easiness or optimized code

[optimize=0]



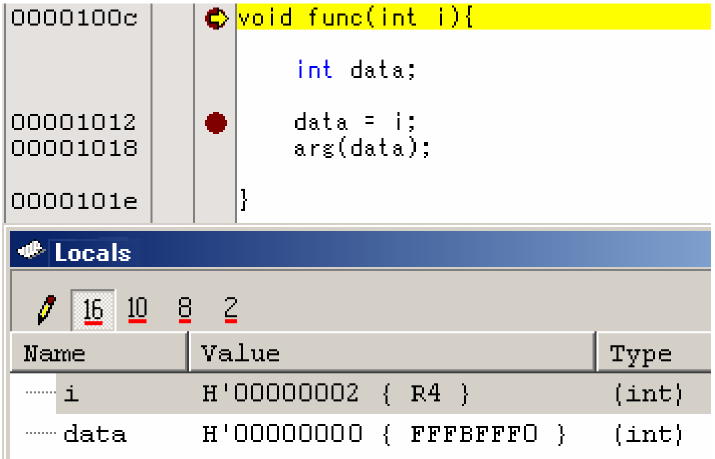
```
0000100c      void func(int i){
                  int data;

00001010          data = i;
                  arg(data);

00001018      }
```

ローカル

Name	Value	Type
i	H'00000002 { R4 }	(int)
data	Not available now.	

Local variables may not be visible in some cases, but code is best optimized. Suitable for final object generation.

[optimize=debug_only]



```
0000100c      void func(int i){
                  int data;

00001012          data = i;
00001018          arg(data);

0000101e      }
```

Locals

Name	Value	Type
i	H'00000002 { R4 }	(int)
data	H'00000000 { FFFBFFF0 }	(int)

Local variables are always visible, but code is NOT best optimized. Suitable for algorithm debug.

MISRA-C rule checker

❑ MISRA C is a set of software development guidelines for the C programming language developed by MISRA (Motor Industry Software Reliability Association). Its aims are to facilitate code safety, security, portability and reliability in the context of embedded systems. MISRA has evolved as a widely accepted model for best practices by leading developers in sectors including aerospace, telecom, medical devices, defense, railway, and others. There is also a set of guidelines for MISRA C++.

❑ MISRA C is not an open standard; the guideline documents must be bought by users or implementers.

❑ SQMlint is a tool to inspect C source codes according to MISRA C rules. Because SQMlint assists developers in source code review through automatic inspection, it helps to develop high-quality C source codes efficiently. Furthermore, because improper codes can be detected by only adding compile options, the user can correct such codes easily when correcting source code pointed by compiler error messages.

MISRA-C rule checker

- ❑ SQMlint is a tool to inspect C source codes according to MISRA C rules. Because SQMlint assists developers in source code review through automatic inspection, it helps to develop high-quality C source codes efficiently. Furthermore, because improper codes can be detected by only adding compile options, the user can correct such codes easily when correcting source code pointed by compiler error messages.

- ❑ Inspection is possible by only adding options when compiling source files.
  - o Source files can be inspected at the same time they are compiled.
  - o There is no need to build an inspection-purpose environment.
  - o Adaptable to special options of the Renesas C compiler.
- ❑ Can inspect in about less than half the compile time.
- ❑ Does not affect the compile results at all.
- ❑ Usable in Renesas IDE: HEW, CS+, e2 studio

MISRA-C rule checker

❑ MISRA C rule inspection during compilation

- Source codes can be inspected against MISRA C rules by only specifying options when compiling the source files.

❑ Adaptable to special options of the Renesas C compiler

- If special options such as one that handles the double type as float type are used, C source codes are inspected by taking the effects of those options into consideration.

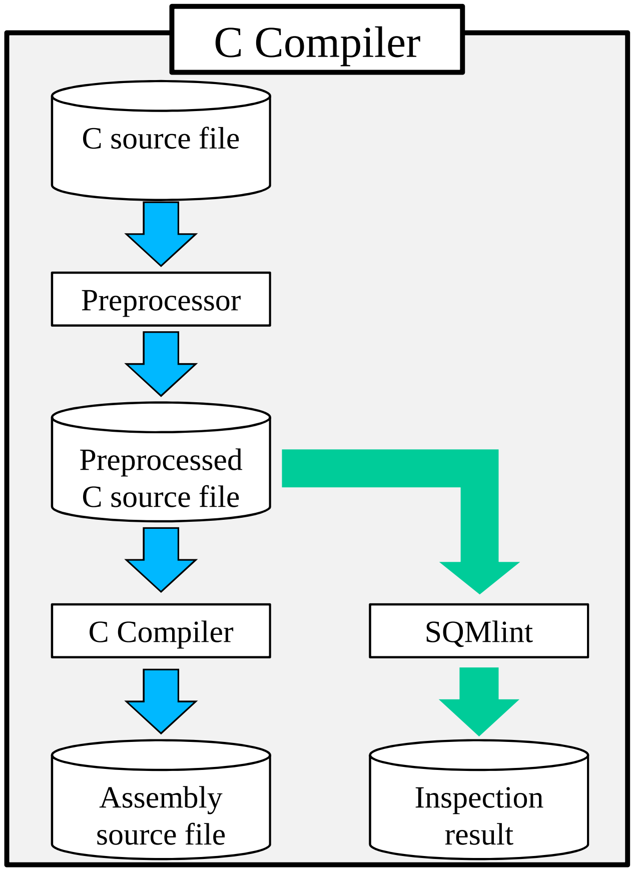
❑ Usable in Renesas IDE

- You can make a tag jump in Renesas IDE on the output results and correct the C source codes on the spot.
- Therefore, deviations from MISRA C rules can be corrected in the same way as when correcting compile errors.

❑ Inspection results output in CSV format

- The Inspection results are output in CSV format usable in spreadsheet software, allowing you to put the inspection results in order easily.

MISRA-C rule checker



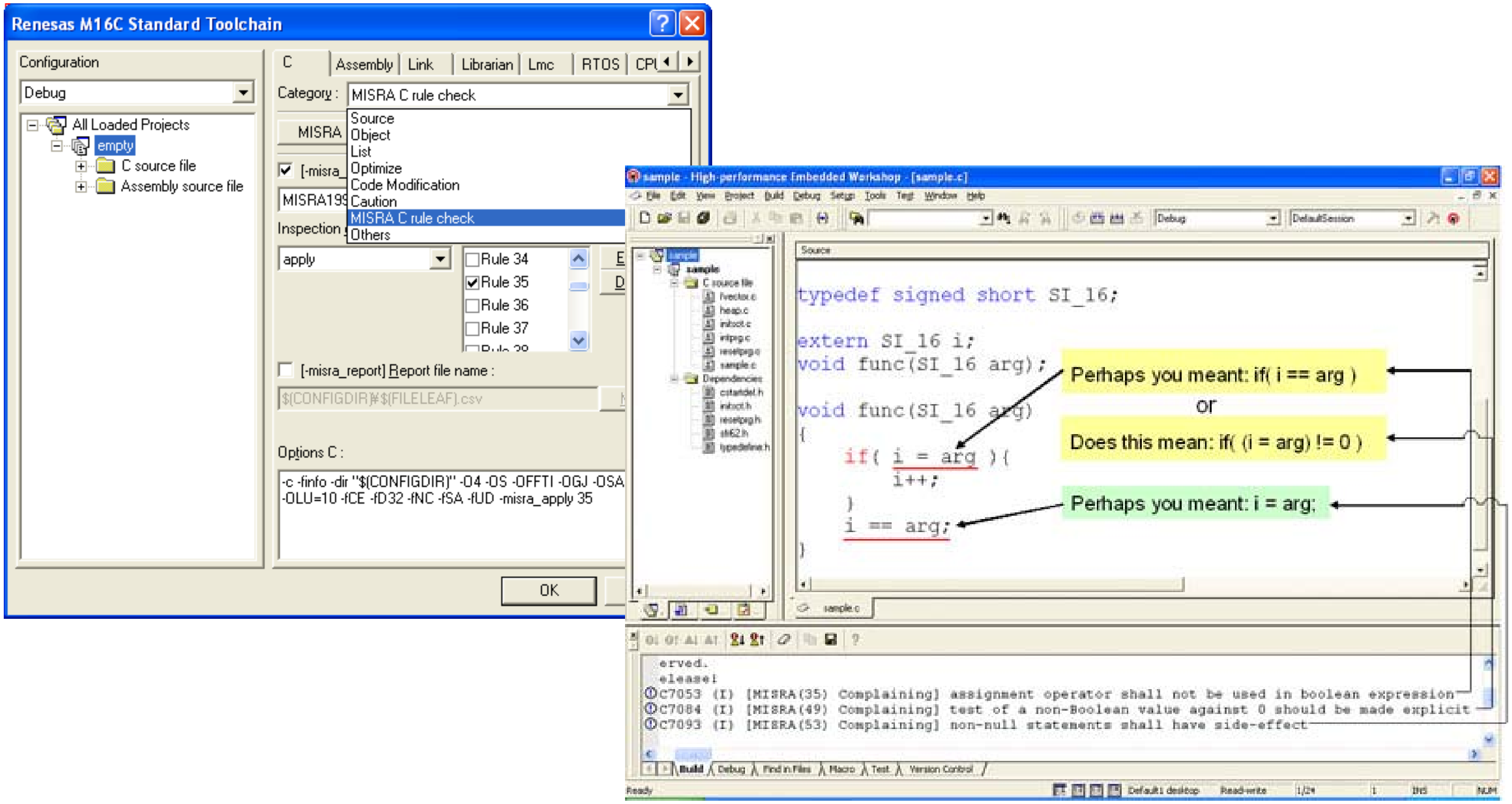
Number of MISRA C Rules which can be Inspected by **SQMlint**

Rule classification	MISRA C 1998	MISRA C 2004	MISRA C 2012
Required	67/93	79/122	[redacted]
Advisory	19/34	13/20	[redacted]
Total	**86/127**	**92/142**	**79/143**

MISRA-C:2012 contains 143 rules and 16 "directives"; each of which is classified logically into a number of categories.

MISRA-C rule checker

SQMlint is integrated in HEW



Configuration: Debug

Category : MISRA C rule check

☑ [-misra_

apply

□ Rule 34
☑ Rule 35
□ Rule 36
□ Rule 37

□ [-misra_report] Report file name :

$(CONFIGDIR)¥$(FILELEAF).csv

Options C :

```
-c -finfo -dir "$(CONFIGDIR)" -O4 -OS -OFFTI -OGJ -OSA
-OLU=10 -fCE -fD32 -fNC -fSA -fUD -misra_apply 35
```

```
typedef signed short SI_16;

extern SI_16 i;
void func(SI_16 arg);

void func(SI_16 arg)
{
    if( i = arg ){
        i++;
    }
    i == arg;
}
```

Perhaps you meant: if(i == arg)

or

Does this mean: if((i = arg) != 0)

Perhaps you meant: i = arg;

```
erved.
elease!
C7053 (I) [MISRA(35) Complaining] assignment operator shall not be used in boolean expression
C7084 (I) [MISRA(49) Complaining] test of a non-Boolean value against 0 should be made explicit
C7093 (I) [MISRA(53) Complaining] non-null statements shall have side-effect
```

Simulation tools

- ➢ Simulator outline
- ➢ Simulator features

Simulator outline

- ❑ The Renesas simulator enables you to debug/test software modules without target hardware. Useful debug function is provided, such as go/break, step, coverage and interrupt handler debug support.

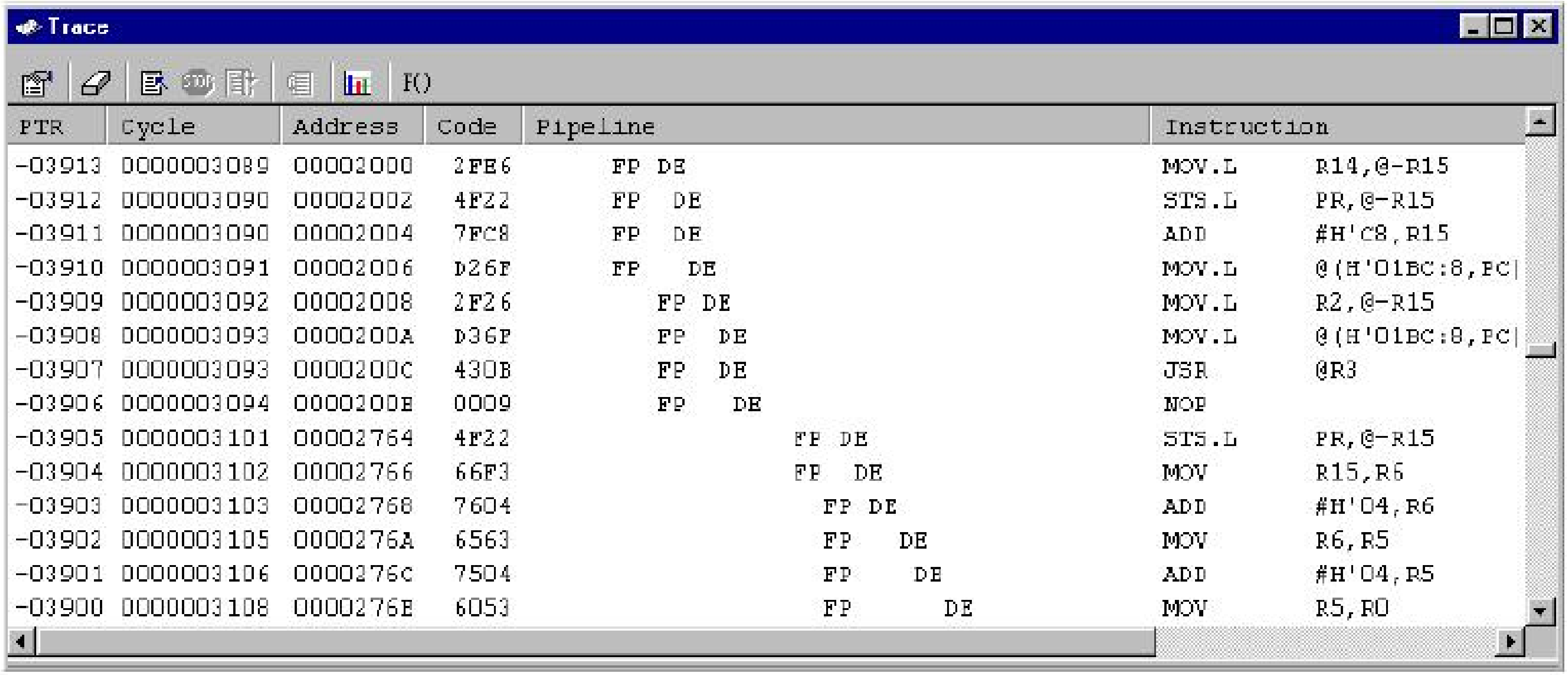
Simulator features

❑ The Renesas simulator enables you to debug/test software modules without target hardware. Useful debug function is provided, such as go/break, step, coverage and interrupt handler debug support.

- Support for ELF/DWARF object format
- Support graphic display of execution cycle count and called count in function units
- Provide stack trace function
- Display cache hit rate
- Display state of pipeline
- Provide extensive break point functionalities (Pseudo interrupt is available)
- Display coverage information in assembly source level
- Provide debug function in visual display with images and waveforms
- Display memory access counts in memory type units (embedded-memory or external memory)

HEW Simulator Window Example

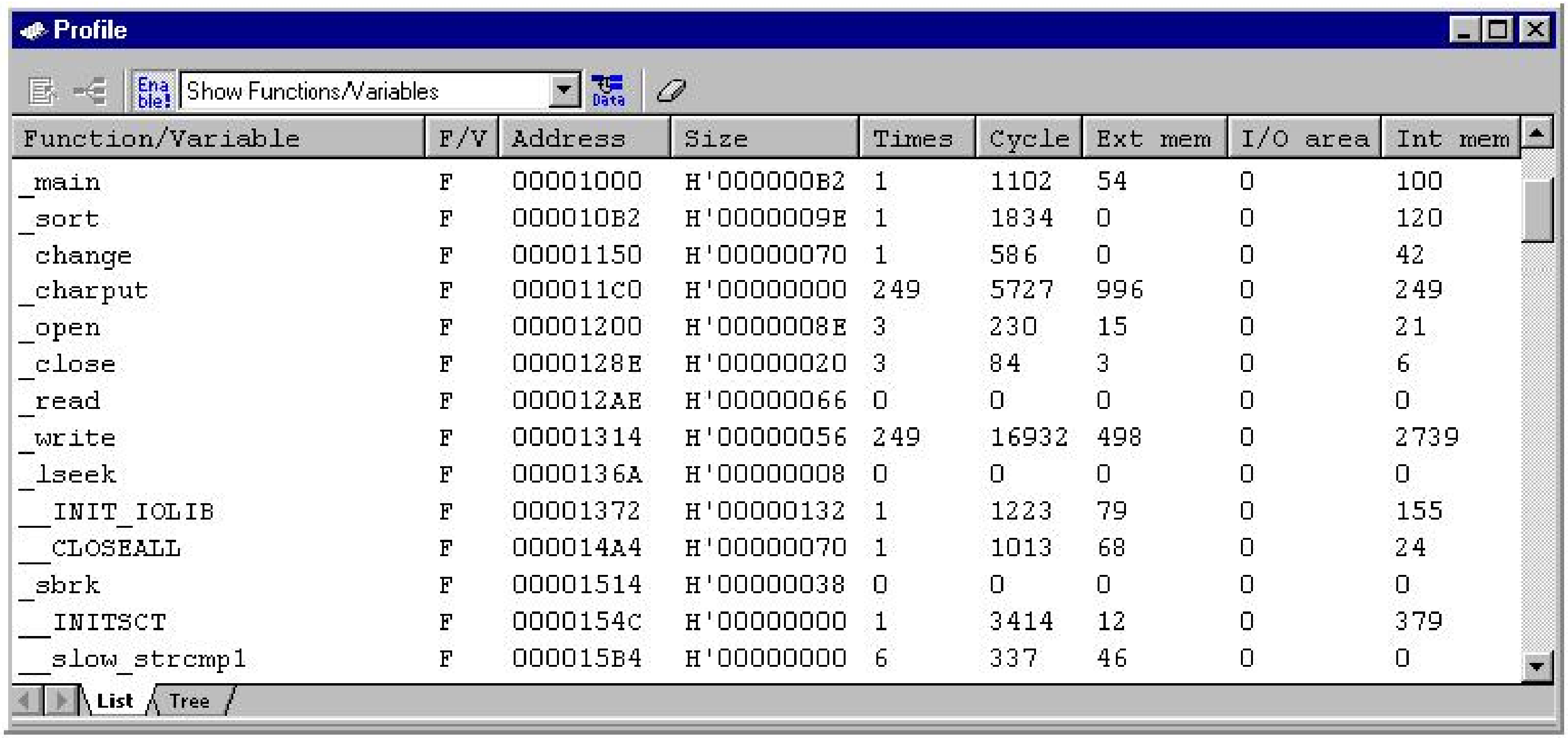
Trace window with pipeline execution status



```
PTR     Cycle       Address   Code   Pipeline                         Instruction
-03913  0000003089  00002000  2FE6       FP DE                        MOV.L    R14,@-R15
-03912  0000003090  00002002  4F22       FP  DE                       STS.L    PR,@-R15
-03911  0000003090  00002004  7FC8       FP  DE                       ADD      #H'C8,R15
-03910  0000003091  00002006  D26F       FP   DE                      MOV.L    @(H'01BC:8,PC|
-03909  0000003092  00002008  2F26          FP DE                     MOV.L    R2,@-R15
-03908  0000003093  0000200A  D36F          FP  DE                    MOV.L    @(H'01BC:8,PC|
-03907  0000003093  0000200C  430B          FP  DE                    JSR      @R3
-03906  0000003094  0000200E  0009          FP   DE                   NOP
-03905  0000003101  00002764  4F22                   FP DE            STS.L    PR,@-R15
-03904  0000003102  00002766  66F3                   FP  DE           MOV      R15,R6
-03903  0000003103  00002768  7604                     FP DE          ADD      #H'04,R6
-03902  0000003105  0000276A  6563                     FP   DE        MOV      R6,R5
-03901  0000003106  0000276C  7504                     FP    DE       ADD      #H'04,R5
-03900  0000003108  0000276E  6053                     FP      DE     MOV      R5,R0
```

HEW Simulator Window Example

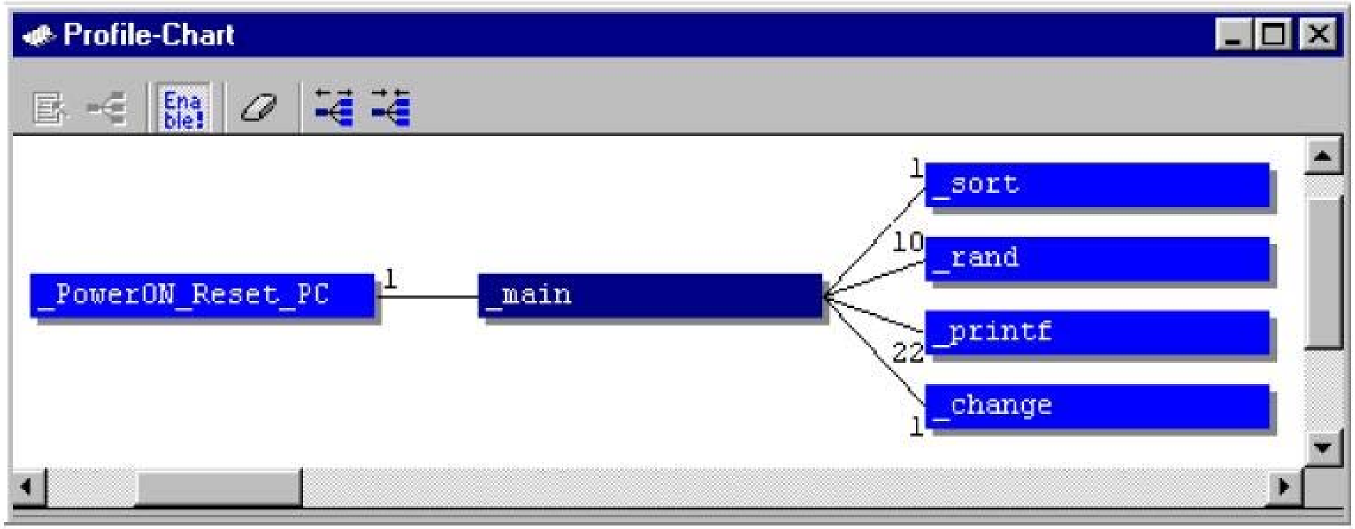
Profile window after execute the program by simulator



Function/Variable	F/V	Address	Size	Times	Cycle	Ext mem	I/O area	Int mem
_main	F	00001000	H'000000B2	1	1102	54	0	100
_sort	F	000010B2	H'0000009E	1	1834	0	0	120
change	F	00001150	H'00000070	1	586	0	0	42
_charput	F	000011C0	H'00000000	249	5727	996	0	249
_open	F	00001200	H'0000008E	3	230	15	0	21
_close	F	0000128E	H'00000020	3	84	3	0	6
_read	F	000012AE	H'00000066	0	0	0	0	0
_write	F	00001314	H'00000056	249	16932	498	0	2739
_lseek	F	0000136A	H'00000008	0	0	0	0	0
__INIT_IOLIB	F	00001372	H'00000132	1	1223	79	0	155
__CLOSEALL	F	000014A4	H'00000070	1	1013	68	0	24
_sbrk	F	00001514	H'00000038	0	0	0	0	0
__INITSCT	F	0000154C	H'00000000	1	3414	12	0	379
__slow_strcmp1	F	000015B4	H'00000000	6	337	46	0	0

HEW Simulator Window Example

Function calls chart after simulation



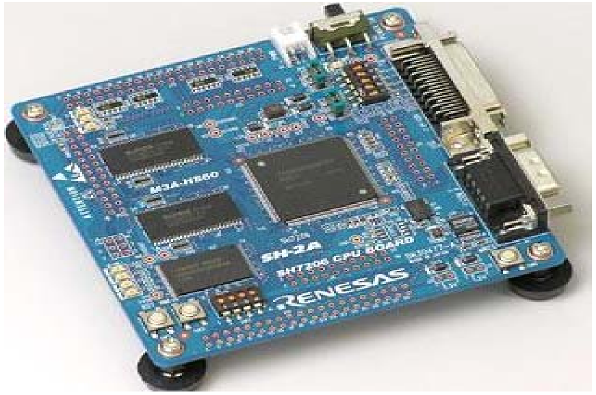
Evaluation boards

Renesas evaluation board

Renesas design and sell the evaluation board for the customers.

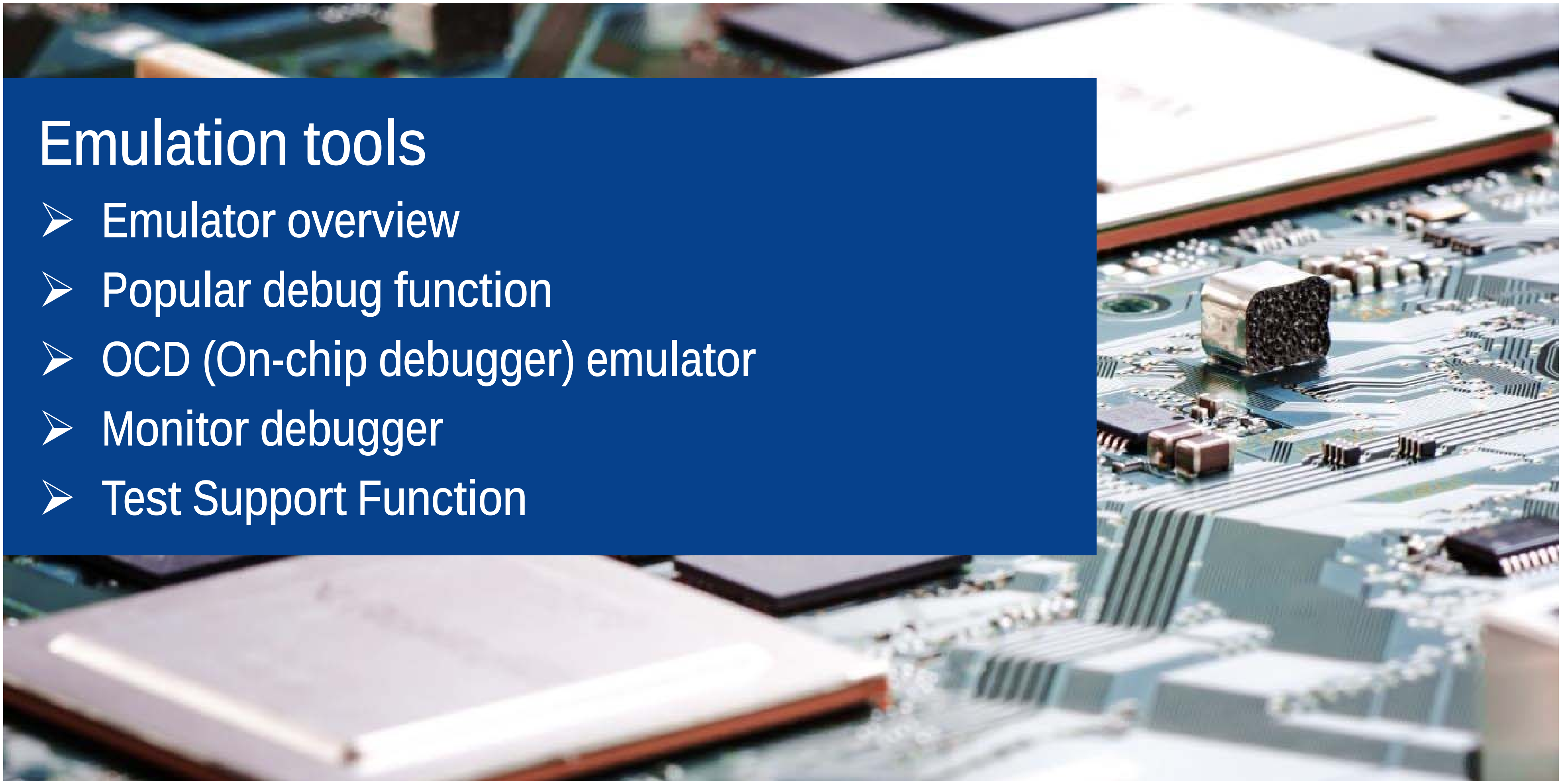
When Renesas start new product promotion, we recommend Renesas evaluation board, because this is available earlier than other board from other company.

Renesas evaluation board include the minimum hardware and no monitor program.
When you use, E10A-USB or other H-UDI interface emulator is required.



SH7206 evaluation board

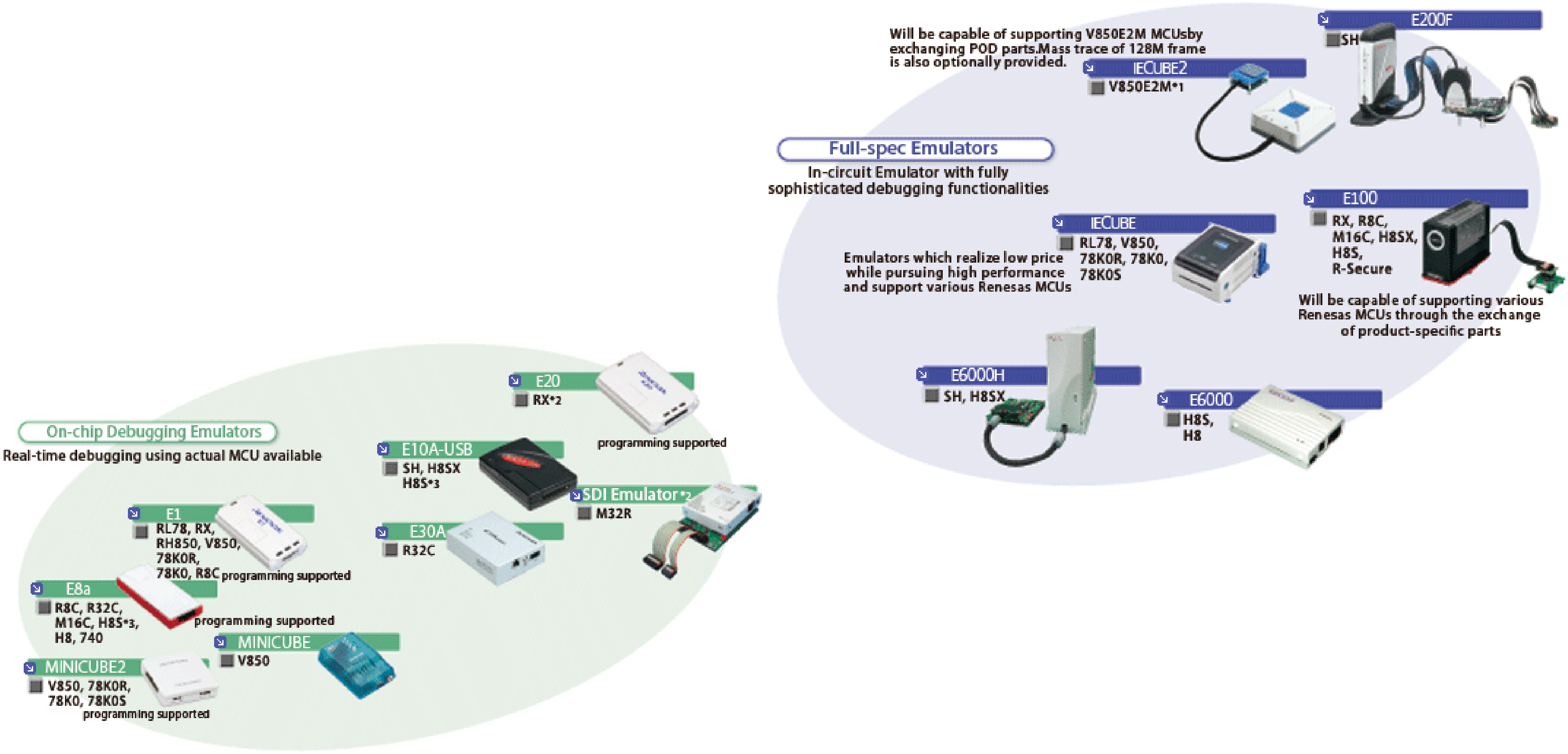
R-Car evaluation board



Emulation tools

- ➢ Emulator overview
- ➢ Popular debug function
- ➢ OCD (On-chip debugger) emulator
- ➢ Monitor debugger
- ➢ Test Support Function

What is Emulator?



Full-spec Emulators

In-circuit Emulator with fully sophisticated debugging functionalities

Will be capable of supporting V850E2M MCUsby exchanging POD parts.Mass trace of 128M frame is also optionally provided.

Emulators which realize low price while pursuing high performance and support various Renesas MCUs

Will be capable of supporting various Renesas MCUs through the exchange of product-specific parts

On-chip Debugging Emulators

Real-time debugging using actual MCU available

Emulator vs. Simulator

❑ What is Emulator’s advantage against Simulator?

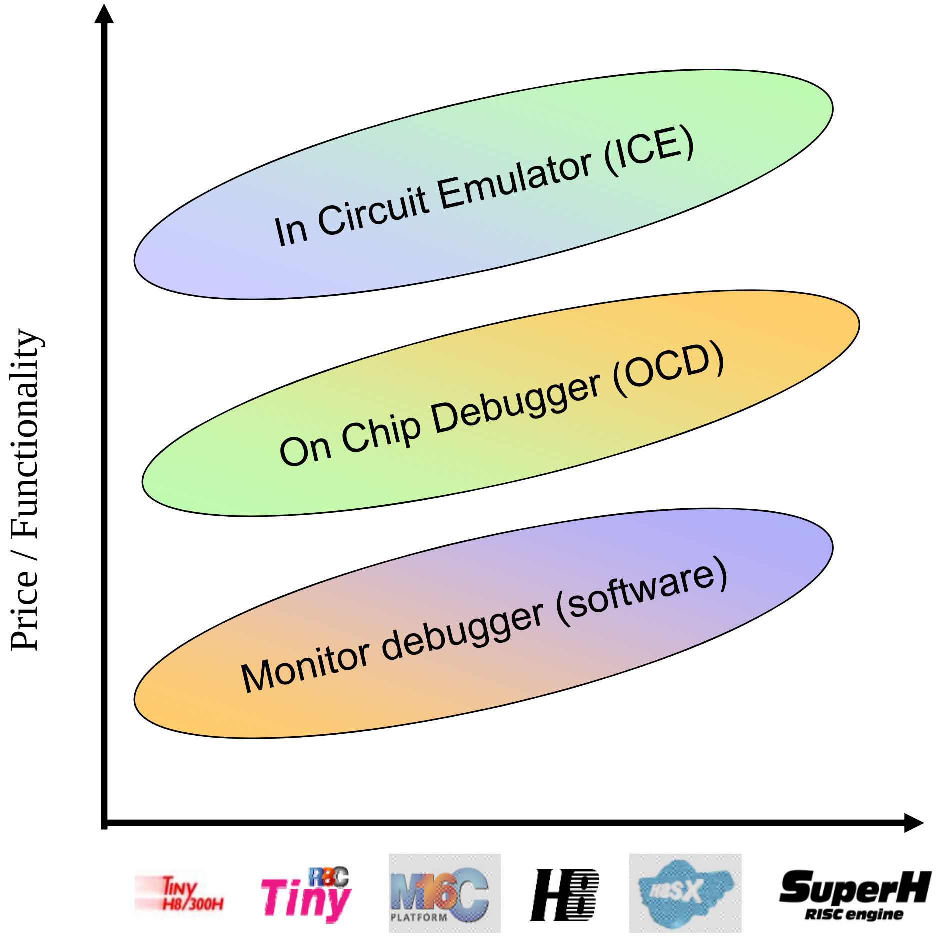
- o Debug with target hardware
- o Software for physical layer access, such as driver, ipl, etc.
- o Combinational bus operation by CPU and DMA (or any other bus master).
- o Interrupt collision detection
- o Non synchronous communication between two different hardware.
- o Any other timing critical scenario

❑ What is Simulator’s advantage against Emulator?

- o Debug without target hardware (before target hardware ready)
- o Low cost
- o Precise CPU internal view, such as CPU pipeline operation view
- o Concentrate on software debug, independent from hardware bug

Emulator products

- ❑ Debug tool who provides debug function with target hardware, is divided into three (3) types of product.
  - o **ICE (In-circuit emulator)** is high functional/high price debug tool.
  - o **OCD (On-chip debugger)** is low cost ICE with reduced debug function. Currently most popular debug tool in the world.
  - o **Monitor debugger** is lowest price debug solution by software. Lower debug capability than ICE/OCD.



Emulator products – Differences

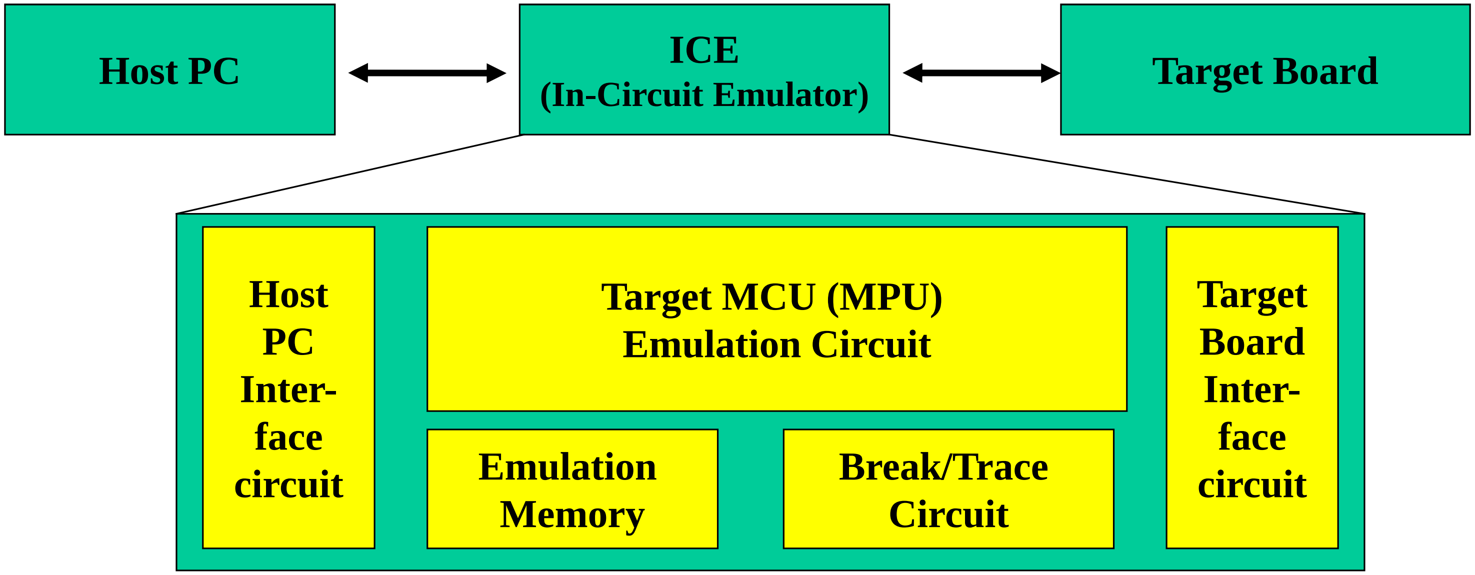
- ❑ ICE (In-Circuit Emulator)
  - o Traditional debug tool
  - o High price, high functionality debug tool
  - o Target interface : IC socket on the target board
  - o Due to CPU clock speed up, it becomes hard to develop ICE for high-end MPU, such as SH-4(4A), RH850
- ❑ OCD (On Chip Debugger)
  - o Mid price, mid functionality debug tool
  - o Target interface : JTAG connection (or serial)
  - o Production device includes DM(Debug Module)
- ❑ Monitor
  - o Low price, low functionality debug tool
  - o Target interface : serial (or Ethernet)
  - o Traditional debug tool made by software.
  - o No specific hardware is required, but debug function is limited.

Popular debug function

- ❑ Breakpoint: Function to stop your program execution at the just point you want to stop
  - o **Hardware Breakpoint**: Break function supported by dedicated hardware. In most devices, complex break condition, such as PC+operand address/data+read/write, is supported.
  - o **Software breakpoint**: Break function supported by code replacement. Complex break condition is not supported, just PC break only.

- ❑ Trace
  - o **PC(Program Counter) trace**: Shows back trace log of PC(Program Counter). After the break, you may find that PC is not the value of your expectation. Then you need to find why (and from where) program comes and stops here. Trace function gives you the useful information for analyzing the program flow.
  - o **Data trace** is also available.

ICE



Host PC

ICE
(In-Circuit Emulator)

Target Board

Host PC Interface circuit

Target MCU (MPU)
Emulation Circuit

Emulation Memory

Break/Trace Circuit

Target Board Interface circuit

Sample:

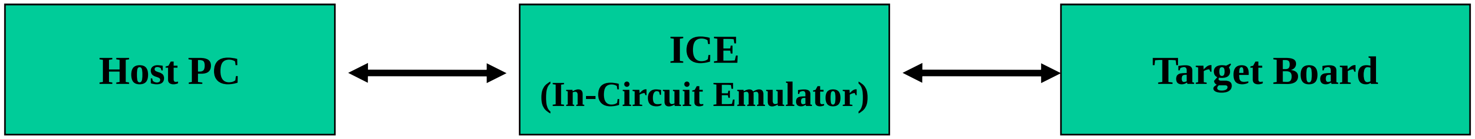
E600H

M32100T5-SDI-E

E100

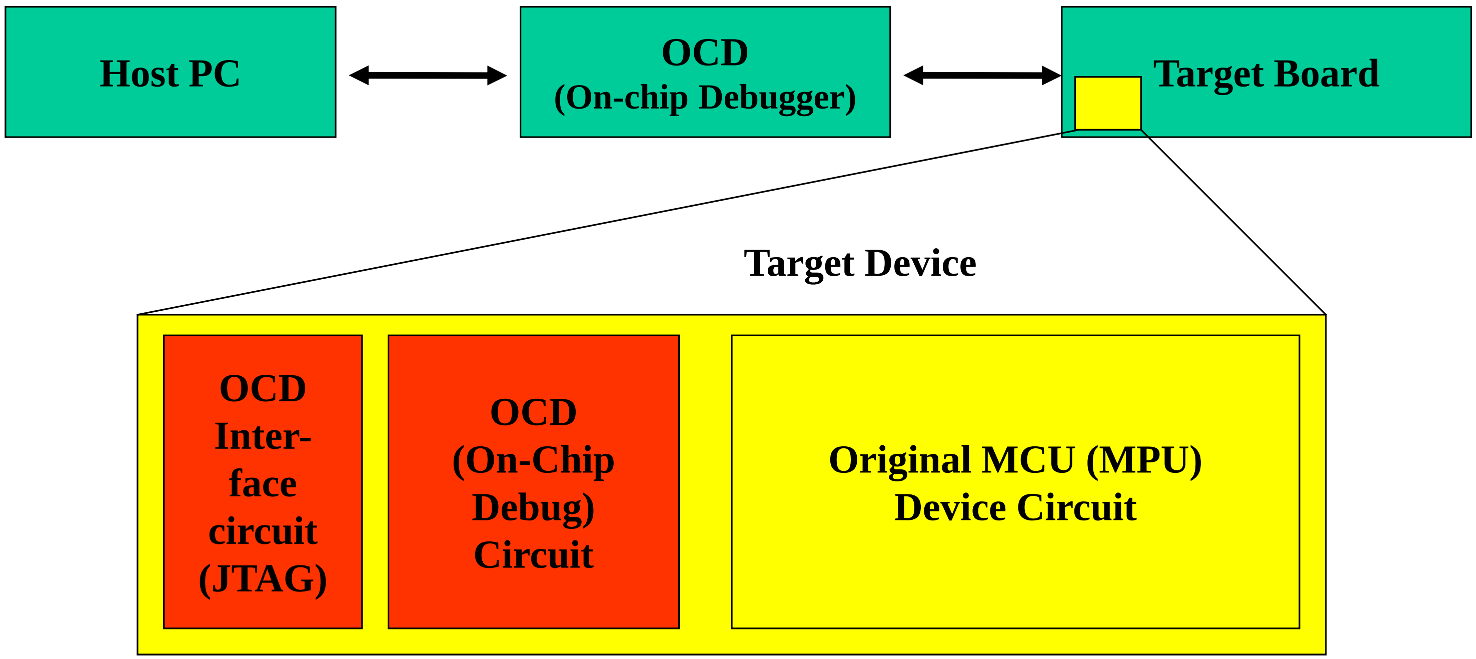
IECUBE2

ICE



- ❑ If the target board does not work correctly, ICE works fine without any problem. As this is independent hardware to target board.
- ❑ ICE has its memory (emulation memory) for target program, it is possible to test software without target board.

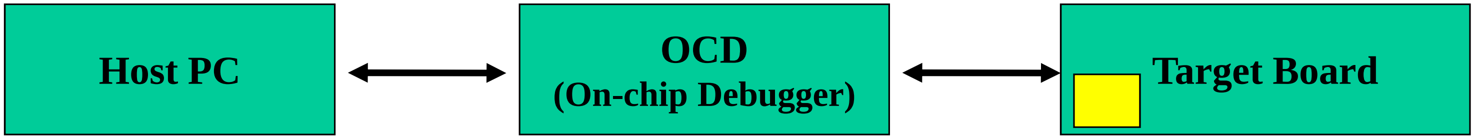
OCD


Host PC
OCD
(On-chip Debugger)
Target Board
Target Device
OCD Inter-face circuit (JTAG)
OCD (On-Chip Debug) Circuit
Original MCU (MPU) Device Circuit

Sample:

RENESAS E1
E10A
RENESAS E20
RENESAS
MINICUBE

OCD



- ❑ If target board hardware has problem, then OCD emulator does not work.
- ❑ There is no way to run target program, if target board does not work.

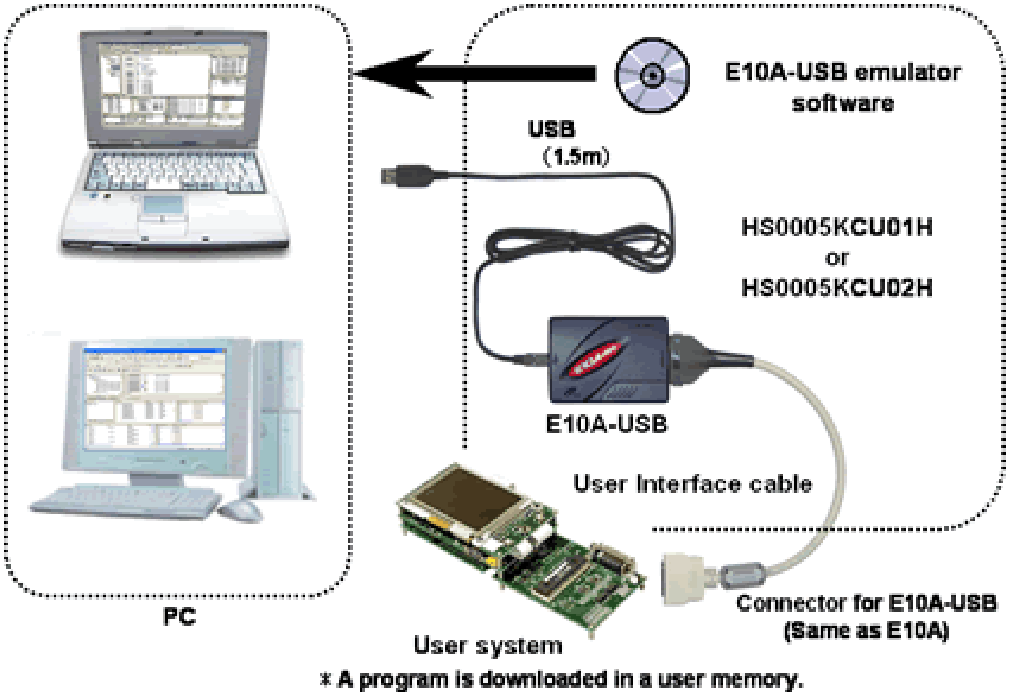
OCD

The OCD emulator offers lower price than ICE. Renesas OCD emulator price is from $50 to $1000.

There are several debug interfaces:

(1) JTAG(H-UDI): SH, H8SX, H8S

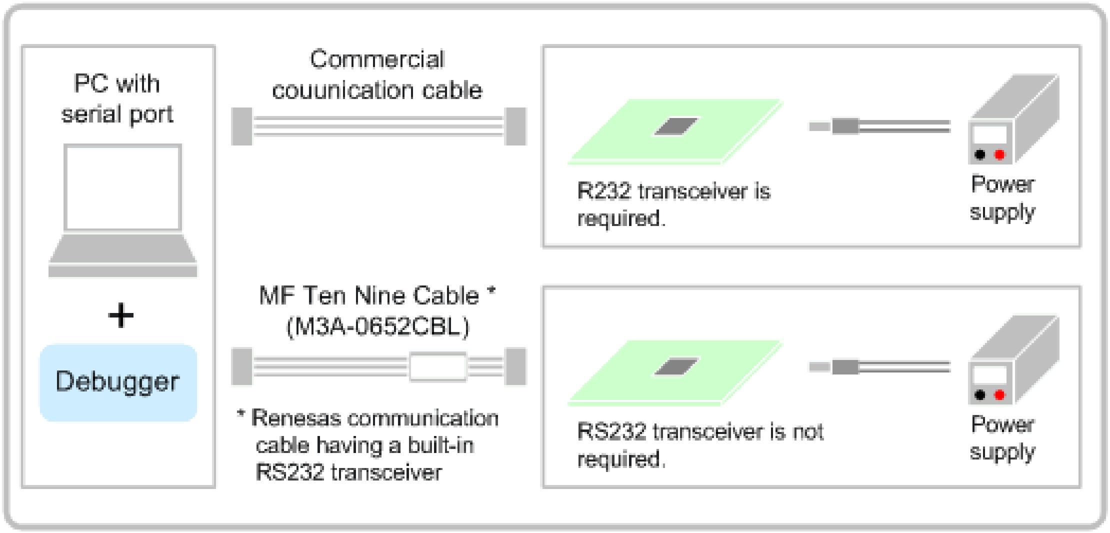
(2) UART:H8/Tiny, H8/SLP, R8C, M16C



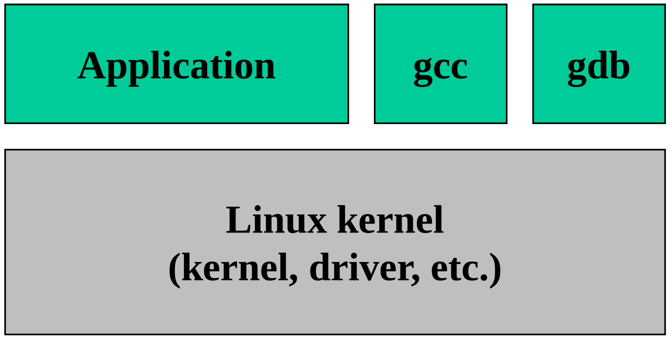
Monitor debugger

Monitor debugger does not need any hardware assistance (except interface cable between target hardware and host PC). After the monitor program is downloaded into the target hardware, the program starts communication to host PC. Most popular interface now is USB.

Monitor program does not work, when communication channel hardware is not working.



Debug function in Linux



On Linux system, debugger (gdb) is one application software.

Gdb works with kernel to provide debug function for application software engineer.

Standard IO (keyboard/mouse and display) is used for debug information communication.

Target hardware must have such standard IO, if debug function is required.

Test Support Function

- ❑ Primary goal of debug is remove bug and make target program work (functional debug).

- ❑ Beyond functional debug, you may need to work further in following two points:
  - o Quality Assurance
  - o Performance enhancement

- ❑ Emulator provides debug function not only for functional debug, but also for software quality improvement and performance enhancement.

Code Coverage

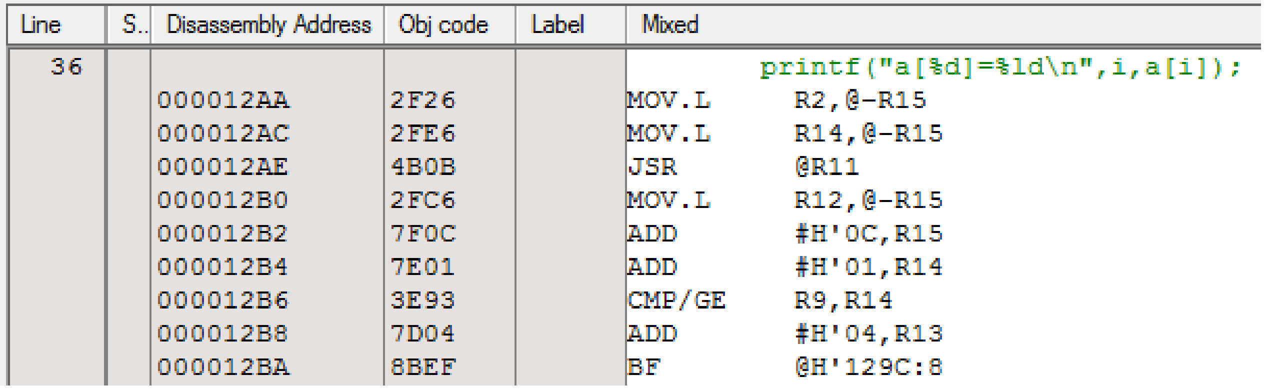
- ❑ Code coverage is one of software quality index in test phase.
- ❑ One important test criteria for high quality software is to achieve 100% C0 and/or C1 coverage.
- ❑ Some emulator supports C0 coverage function.

- ❑ For example, E200F Emulator (for SH-4A) support following coverage function.
  - o C0 coverage (C1 coverage is NOT supported)
  - o 4M Byte address range (default)
  - o Prefetch cancel
  - o Source code view with coverage result (executed / not executed)

C0 and C1 Coverage

- ❑ C0 coverage: How many lines (or instructions) are executed against all source lines (or all instructions).
- ❑ C1 coverage: How may branches are taken (or not taken) against all branch cases

- ❑ Coverage definition in C source level (statement) and assembly level (instruction) is different



Line	S..	Disassembly Address	Obj code	Label	Mixed
36					`printf("a[%d]=%ld\n",i,a[i]);`
		000012AA	2F26		MOV.L R2,@-R15
		000012AC	2FE6		MOV.L R14,@-R15
		000012AE	4B0B		JSR @R11
		000012B0	2FC6		MOV.L R12,@-R15
		000012B2	7F0C		ADD #H'0C,R15
		000012B4	7E01		ADD #H'01,R14
		000012B6	3E93		CMP/GE R9,R14
		000012B8	7D04		ADD #H'04,R13
		000012BA	8BEF		BF @H'129C:8

Example: 1 C statement equivalents to 9 Assembly instructions.

C0 and C1 Comparison

- ❑ C0 coverage 100%, but C1 coverage is NOT 100%

```
          Statement A
          Conditional Branch B
             (if true branch to ‘Lable1’)
          Statement C
Label1:   Statement D
```

If conditional branch ‘B is not taken’ at the test case,
all statement is executed

➔ C0 is 100%.

But, ‘branch B taken’ is not tested in such test case

➔ C1 is NOT 100%

C0 and C1 Comparison

❑ C0 coverage is NOT 100%, but C1 coverage is 100%

```
          Statement A
          Conditional Branch B
             (if true branch to ‘Lable1’)
          Statement C
          Branch always
             (always branch to ‘Lable2’)
          Statement D
Label1:   Statement E
Label2:   Statement F
```

Statement D can NOT be executed in any test case.

➔ C0 is NOT 100%

Statement D must be removed from source code, or need to change source code in proper way.

➔ Do not release any ‘dead’ code in output.

Performance / Profile

❑ Performance enhancement can be achieved by hardware upgrade (takes higher frequency, wider bus, etc.) or software tuning. In high-end application, there are many rooms for performance enhancement by software tuning.

❑ Some emulator provides performance (or profile) function for software performance tuning. This function provides several useful information for performance enhancement, such as execution time information in specific function or specific address range (from start point to end point).

- o CPU clock count (or number of executed instructions) from specific start address to specific end address
- o number of i-cache hit (or number of i-cache miss hit)
- o number of o-cache hit (or number of o-cache miss hit)
- o number of interrupt (or number of exception)

www.renesas.com