To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Development Tool

Effective Usage of Hardware Development Tool (UseDevTool)

Introduction

The objective of this Application Note is to help developers to choose and use the correct hardware development tool.

The Application Note will classify the various hardware tools into different categories, and highlight their pros and cons. The implementation of the tools is also discussed to allow users to have an in-depth understanding.

The various tools for discussion are:

- CPU board, starter kit
- ROM emulator
- JTAG emulator
- Full ICE emulator
- Simulator

Various topics that evolved will be

- Breakpoints
- Trace
- Advanced functions

Although the focus of the Application Note is on Hardware, various topics on software development tools will also be covered briefly to facilitate the explanation.

Target

All

Contents

1. Introduction

The goal of Development Tool is to provide an insight into the MCU, and the necessary control & monitoring means. This is to allow developers to build their application, in which most of the time spent is on fixing bugs. Thus a commonly used term for development tool is debugger or debugging tool.

It may be frustrating for developers to use development tool. They may pay thousand of dollars for an emulator, and spent months to link it to their target system. Worst of all, after months of coding to achieve a full emulation, the downloaded code to an OTP does not guarantee a functional application. Another scenario is to find out the limitations when using a JTAG-like emulator.

On the other hand, Tool support engineers find that 50% of their support activities end up troubleshooting customer's target board. Another 40% of the time handling troubled customers that had misused the tools as they had misunderstood the tools.

All these problems can be avoided. Developers must understand the pros and cons of each tool, and plan their debugging strategies well before the development begin.

This Application Note will help developers to understand different development tools and their implementation & functions.

The various hardware development tools are classified into three main categories.

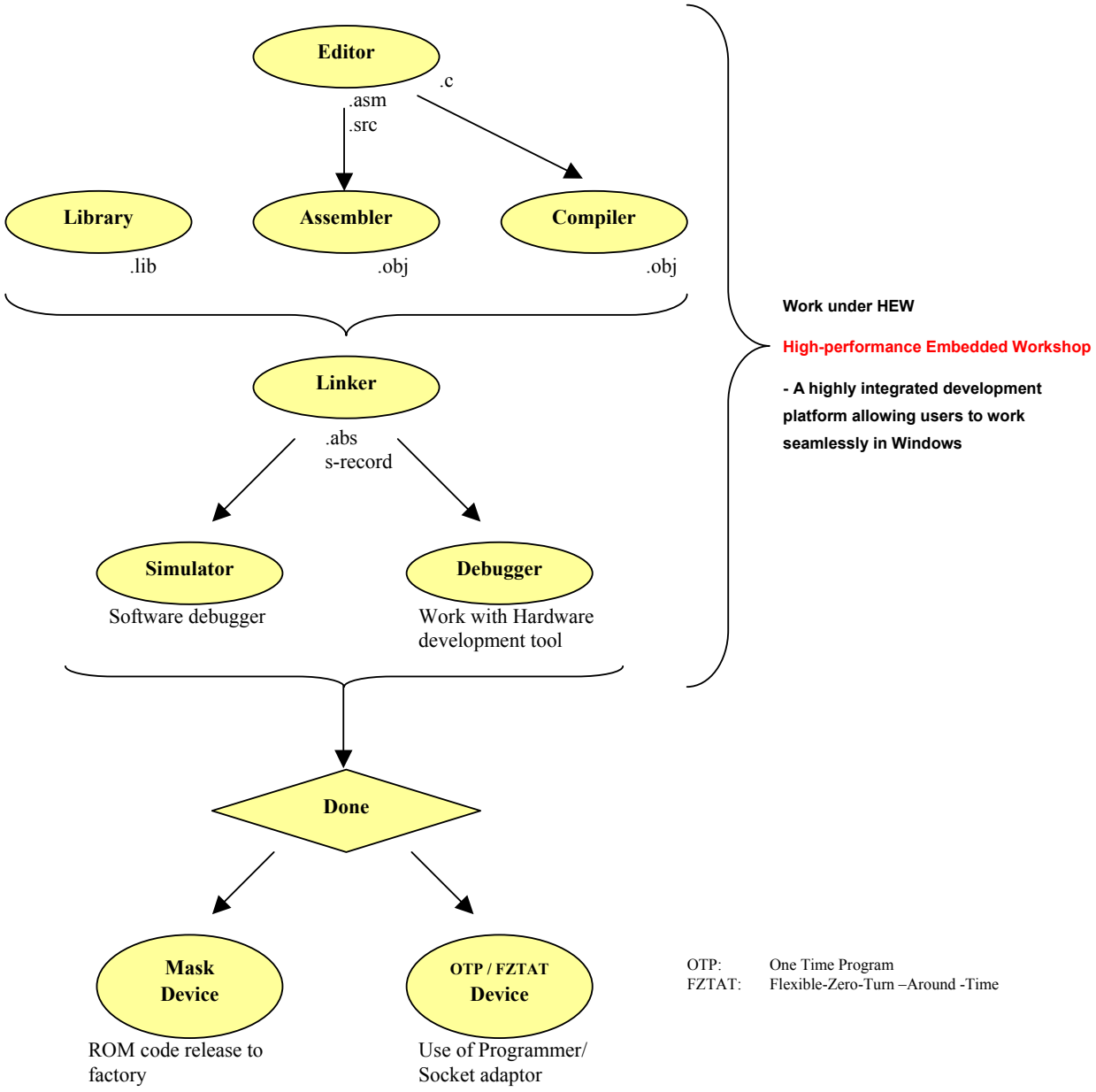
Each category will have the following topics explained:

- General description
- Common Characteristic
- General Implementation Illustration
- Common Pitfalls
- Usages
- Pros & Cons

In the later section, the emulation functions' (such as step, break, Trace and etc.) characteristic in the three ranges of tool will be further elaborated.

2. Development Work Flow

The following give a short overview of development workflow.



Definition

1. **HEW (High-performance Embedded Workshop)**
   - Renesas all-in-one development platform that allow developers to edit, assembler, compile, link, simulate and debug their application seamlessly.
2. **Editor**
   - Text Editor for writing the source code (C or Assembly)
3. **Assembler**
   - Translate human readable code into machine instruction (opcode) -> OBJ file

     e.g. MOV R1, R2 => H'1112
4. **Compiler**
   - Translate High level language (e.g. C) into machine language -> OBJ file
5. **Linker**
   - Bring all OBJ files (output by Assembler and Compiler) together, and organize them into a single manageable file

     -> MOT, ABS file
6. **Debugger** (Used with Hardware Development Tool- e.g.E6000, E7, CE, ALE and etc)
   - Load the Linker output file to the hardware development tool, and perform emulation and debugging such as step, break and trace…
7. **Simulator**
   - A "software emulator" used to model the actual MCU behaviors.
   - Execute the Linker output file on the PC instead of the actual MCU
8. **Flash Writer / Programmer** (Used with Hardware Development Tool)
   - A tool used to write Linker output file to a Flash MCU

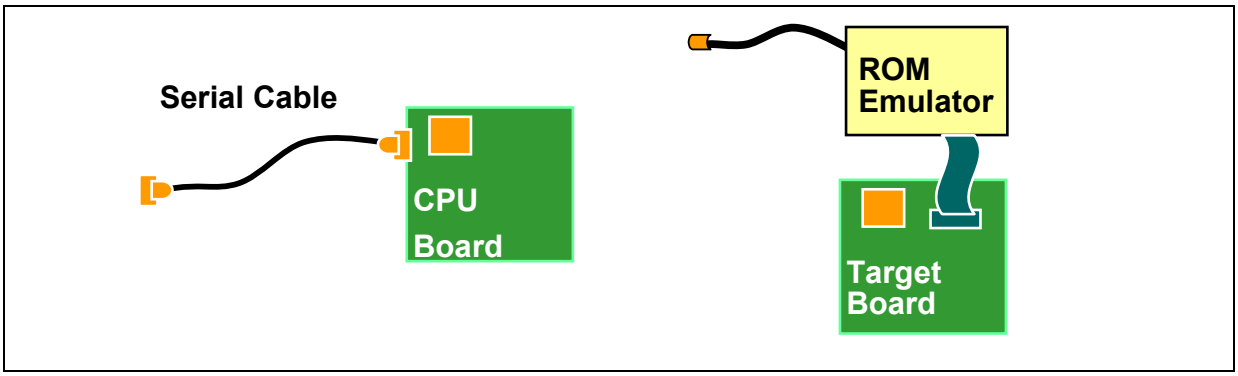
     e.g. FDT (Flash Development Toolkit)
9. **Other Utilities**
   - Call Walker (HEW built-in)
   - Map Viewer (HEW built-in)
10. **Socket Adaptor**
    - An adaptor used to accept the MCU package and can be plugged to most commercially available programmer.

3. Hardware Development Tool Classification and Comparison

There are a numbers of definition and naming in the market. Based on these tools characteristic, the hardware development tool can be divided into three classifications;

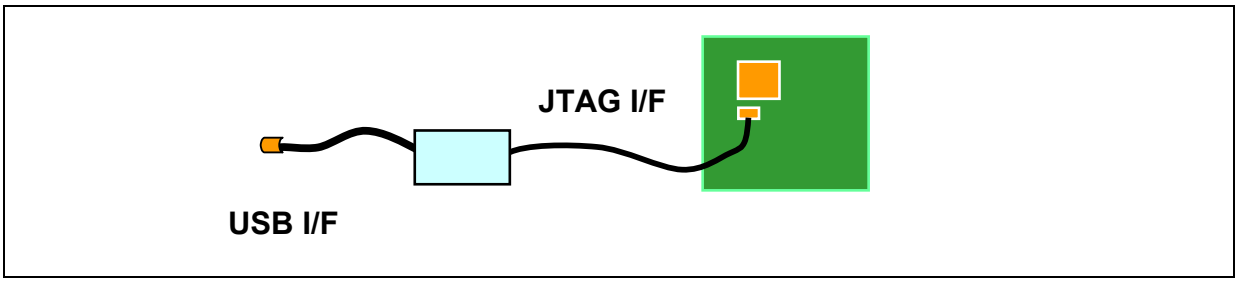
1. CPU board, Starter kit, evaluation board, ROM emulator → **CPU board**

A basic board with the actual CPU mounted on, a software monitor code will be built-in to communicate with the PC for simple debugging.



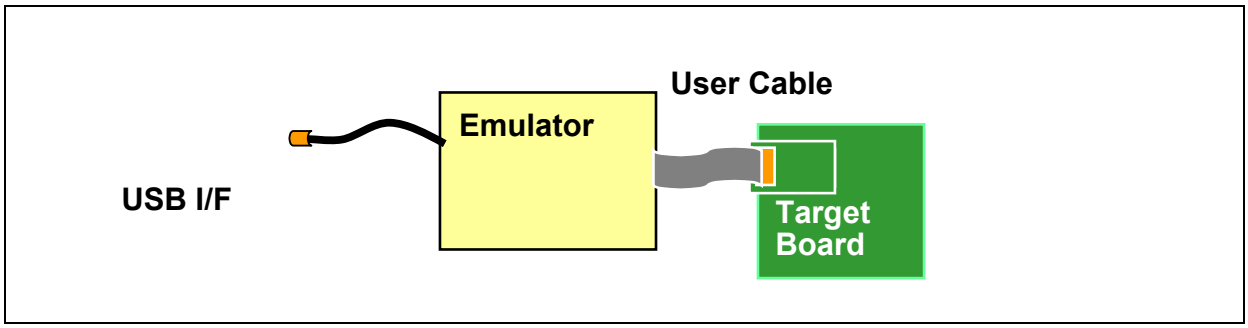
2. JTAG emulator, On-chip Debug, BDM, N-wire → **JTAG-like**

The device must have on-chip debug features in order to communicate to the "JTAG-like" device. Normally connected via some simple wires to perform emulation.



3. ICE (In –Circuit Emulator) → **ICE**

A complex system that carries an evaluation chip (bond-out chip), that can simulate the real chip realistically. Such system can have more advance control & monitoring features. It needs a dedicated user cable to plug onto the user target system.



The following tables show the “general” characteristic of tools:

Type	CPU board	JTAG-like	ICE
Chip Resources	Used partially	Not Used	Not Used
Actual Chip	Used	Used	Not Used
Emulation Function	Basic [30%]	Moderate [50%]	Complicated [100%]
	Load/Run/Edit/ Break	Load/Run/Edit/ Break/ Trace	Load/Run/Edit/ Break/ Trace on-the-fly-access/ complex Break & Trace /Time measurement…
Usage	Basic System, Benchmarking	Moderate complexity system	Complicated system development
Cost	Almost Free	Low Cost	High Cost
Tool Time-to-Market	Fast, easy to develop	Fast	Slow, Difficult to design
Remark	Can be ported to actual application board. Free Tool	Good economical basic tool that guarantee operation (unlike ICE)	An essential tool for high-end applications especially when the project is started from scratch.
Example	SLP CPU board, BAG…	E7, E10	E6000, CE, ALE

The “general” term is used as there are exceptional cases such as:

- Developers may used the CPU board concept to develop a RTOS project,
- Some JTAG-like emulator has no trace function
- Some JTAG-like emulator is more costly than an ICE

The emulation functions of these tools can be further elaborated as follow:

Features	CPU board	JTAG-like	ICE
Download speed	Slow (mostly usage of serial)	Slow (usage of serial interface)	Fast (usage of USB, PCI…)
Step	Simulated step	Control step	Control step
Trace	No Trace	Limited trace (e.g 4 Jump trace address)	Full control of Trace (bus, filter, sub-routine, time stamp data…)
Hardware Break	Dependent on device availability of UBC (User Break Controller)	Limited hardware break	Full control of Break (complex, condition…)
Emulation control	No	No	Guarded area access inhibit, write protection.
Time measurement	No	No	Yes
Possible chip resources used	Usage of serial port, or NMI…	Usage of multiplex port pins, or memory resources	No
Target	Itself is a target	Need a target board with actual chip	Can work without a target. Provision of optional target memory when target is not available.
On-the-fly access	No	No	Yes
Advanced functions	No	No	Coverage, Performance Analysis, Pinview

4. Tools Implementation and Features

4.1 CPU board, Starter kit, Evaluation board, ROM emulator

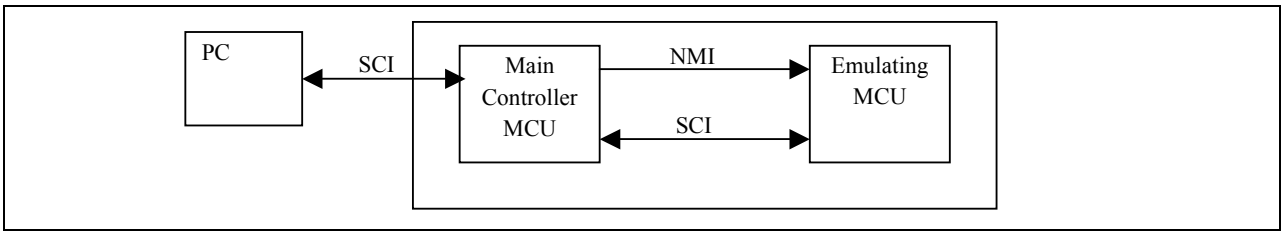
4.1.1 General Description

A basic board with the actual CPU mounted on, a software monitor code will be built-in to communicate with the PC for debugging.

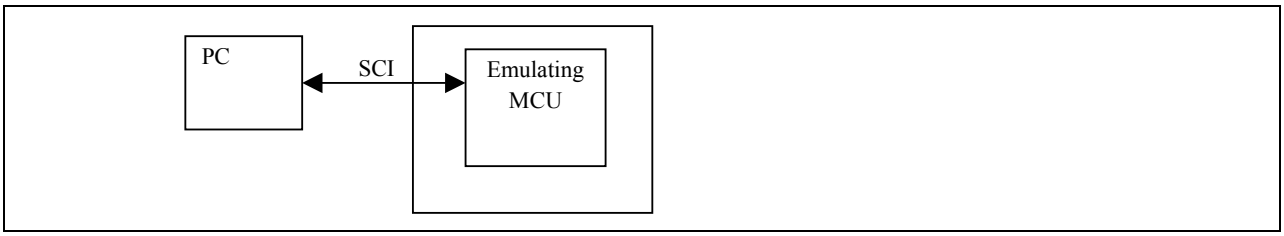
4.1.2 Common Characteristic

Generally this classification of tool uses the actual chip to perform the emulation. Monitor code and serial interface are used to communicate with the PC. Depending on the implementation, the CPU board can be a powerful and user-friendly tool. The SLP CPU board (H8/38024) is an example. The monitor program is communicating with the Renesas HEW. Thus a C-Source level of debugging is possible with the PC GUI.

There are various methods of creating the PC control methodology to the CPU board.



One example is to use another MCU to be the master controller to communicate with the PC, and generate a NMI interrupt to the emulating CPU. In this design, the NMI of the emulating CPU is used (The serial port of the emulating MCU is used to obtain command & data from the PC).



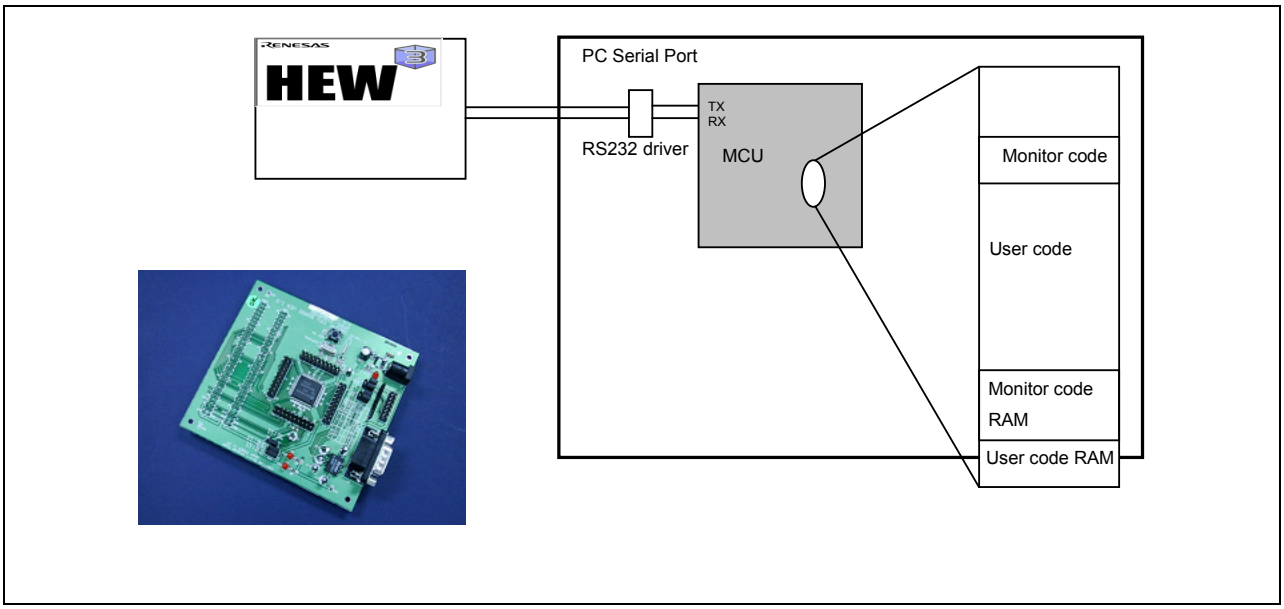
Another implementation is to directly use the emulating MCU serial port's interrupt to perform the communication. Thus user program shall not disable the interrupt, if otherwise the PC will not be able to communicate with the MCU.

The main drawback is the resource usage, such as memory and serial port. This will not have any effect if the application does not require these resources. A possible requirement will be to allow the monitor code to be re-locatable, so that developers have a better control on their application. A normal monitor code will take up 3K of Flash space. If the single step function is removed, developers can have it reduced to 1K of flash space.

Many MCUs do not come with an on chip break controller. However Renesas TINY has a built-in UBC (User break controller). Developers can make use of this UBC to enhance the debugging capability of the CPU Board. The UBC is capable of detecting address and data value so as to create a break condition

The ROM emulator is grouped in this classification as their implementation & features are similar.

4.1.3 General Illustrated Implementation



4.1.4 Common Pitfalls

Common problems faced by developers are:

1. Interrupt handling

During debugging with the SLP CPU Board, HEW will constantly probe the CPU board to check if it has gone into “break mode” (Breakpoints have been set). If the CPU board is running a higher priority interrupt for a longer period of time, HEW will issue a timeout error.

2. Source

In some cases, when the external memory may not be a RAM, but a writable device like flash, which need to have a writing algorithm. The user may wish to ride on the PC GUI to access this external memory. Thus CPU source code will be required for edit. However the monitor code is usually not provided in source but in library format or S-record format.

3. Limited debugging function

There is no trace, unless this is implemented by software means.

4. Co-exist with OS

It will be difficult to port into an OS based application. This is mainly due to the fact that both modules (monitor code & OS) are trying to take full control of the system. Thus “time out“ and “Resource” conflict are encountered. However this implementation is possible if the monitor code is written to be a module of the OS.

4.1.5 Possible Usages

1. Immediate evaluation of peripherals.

- Since there are no other components on board, the CPU board is good for evaluation of peripherals such as PWM, complex timer, ADC and etc

2. Porting to actual application board for basic debugging

- CPU board requires only a serial port. Developers may implement:
    i. Place a serial RS232 driver and connector on-board, to communicate with the PC.
    ii. Place a 4 pins connector (VCC, GND, TX, RX) on board, and built the serial driver externally to save board space & power.

3. Porting to actual application board for Parameter tuning

- Some analog applications may need to fine tune some of the parameters after the whole system is setup in the production floor

4. Benchmarking

- Run the Benchmarking software and measure the runtime through external means (toggling of port pin)

4.1.6 Pros and Cons

Pros

- Low cost / Free
- Provide easy evaluation
- Provision of on-field debugging
- Provide as a secondary tools in big project group
- Support high-speed operation (run with actual) MCU

Cons

- Limited debugging (no trace, no Hardware break…)
- Usage of User resources (ROM, serial port…)
- Cycle steal CPU time for monitoring purposes
- Can only work with Flash devices
- Can be crashed if programmer did not take care of monitor existence

4.2 JTAG emulator, On-chip Debug, N-wire

4.2.1 General Description

The device under test must have on-chip debug features in order to communicate with the “JTAG-like” device. Normally connected via some simple wires to perform emulation.

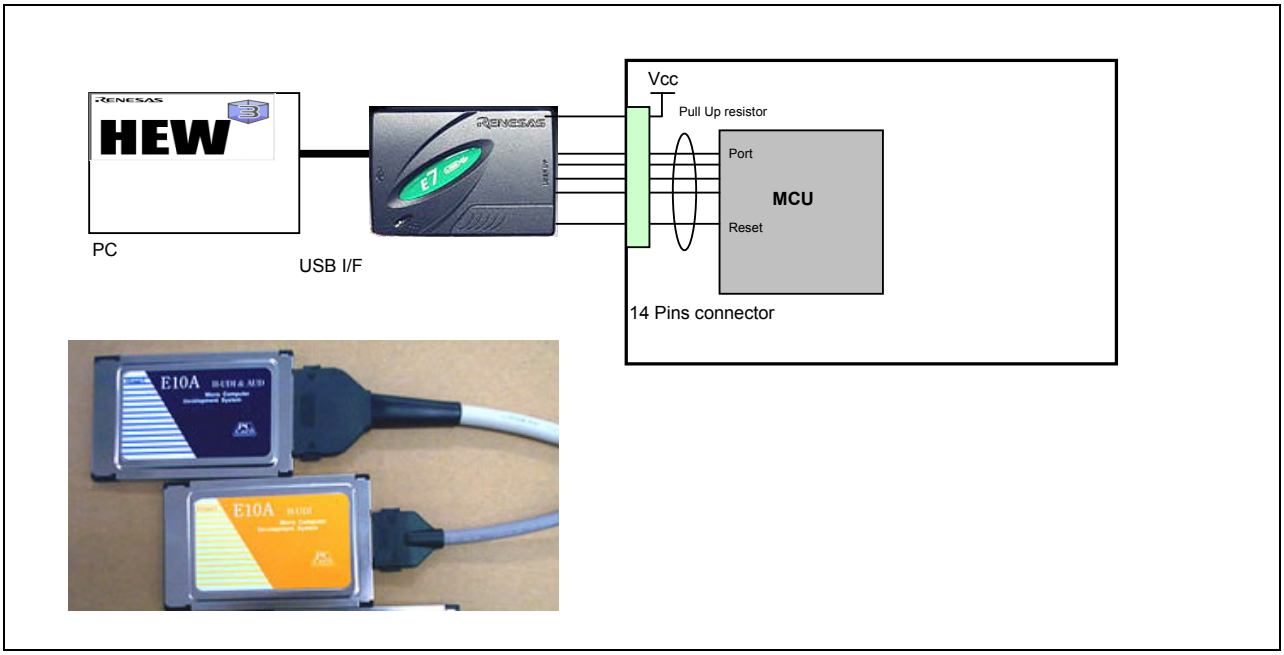
4.2.2 Common Characteristic

This range of tools is possible if the MCU has this on chip capability designed. Generally these tools use a few wires for communication. The control is in serial form, thus limiting the download speed. As such, it is also not possible to have real time trace (too slow to transmit data back to PC), and deep trace record (occupy actual Chip space). The other limitation will be its limited emulation functions. The tool can only achieve basic functions: load, run, step, break and trace. Thus developers need various techniques to help them to identify complex bugs.

However the JTAG-like emulator is still an effective and easy-to-use tool. It has a much compact body, and debugging is simplified by a simple connection of some very reliable connectors. Since the actual chip is used, user is guaranteed a working conditioned application after the debugging.

To overcome the setback of trace-ability, Renesas JTAG-like emulator E10 has an optional AUD interface. This is to capture the external bus cycle of the running processor. This will further enhance the capability of the emulator (Although the internal bus cycle such as cache activities are not captured). A small memory resource is being used by the emulator.

4.2.3 General Illustrated Implementation



4.2.4 Common Pitfalls

Common problems faced by developers are:

- Limited emulation functions
- Not enough trace information.
- Slow download of files

4.2.5 Possible Usages

The JTAG-like emulator can be used in almost all types of development area.

4.2.6 Pros and Cons

Pros

- Low Cost
- Simple & reliable connection
- Support high-speed operation (run with actual) MCU
- Guarantee operation when emulator is removed

CON

- Lack of advance emulation functions (complex Trace, events break…)
- Slow communication speed (Serial)

4.3 ICE (In-Circuit Emulator)

4.3.1 General Description

A complex system that carries an evaluation chip (bond-out chip), that can simulate the real chip realistically. The system can have more advance control & monitoring features. It needs a dedicated user cable to plug onto the user target system.

4.3.2 Common Characteristic

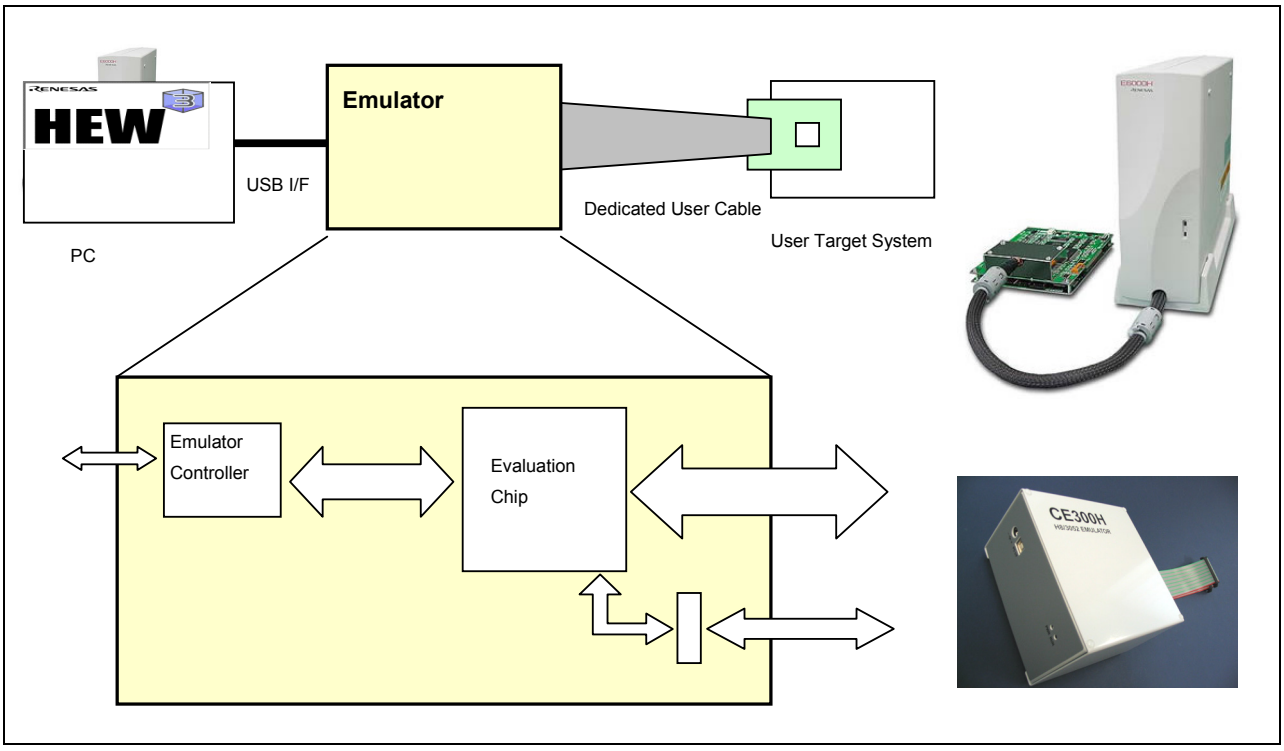
Emulator is the most powerful hardware tool. Due to its complex nature, novice may find it difficult to use. Most complains of the tools will be its cost and target connectivity. Another common observation is that many users do not make full use of the available advance functions.

Inevitably, emulator is an important tool that cannot be replaced, especially in the new project development area. Complex issues will need features such as real time trace, on-the-fly facility, event breakpoints to control the situation. These cannot be achieved with any other ranges of tool.

In order to have a control emulation, emulator takes full control of the whole target. For example, the emulator will be able to mask off the reset, control the clock, access the memory on-the-fly, having software breakpoint and etc. If these implementations are not understood, it will cause unknown issue such as no external watchdog reset, running on wrong clock frequency, non-real time execution, not able to break!

In recent years, MCU operation speed has increased 10 folds, from the 4 MHz ranges to 400MHz ranges. Moreover the launching of new devices has also been becoming faster and faster. These trends, which determined the emulator's design difficulty and its time-to-market requirement has given these tools a negative impact.

4.3.3 General Illustrated Implementation



4.3.4 Common Pitfalls

Common problems faced by developers are:

1. Actual MCU do not work with the emulated code
2. User cable inconsistence in connectivity
3. Heavy & bulky User cable
4. No oscillating clock from external
5. No NMI, reset from target
6. Signal floating
7. Different performance with actual MCU
8. Cannot break

4.3.5 Possible Usages

Emulator can be used in all applications. However in considering of cost and effectiveness, the emulator is best used for full development of a new complex project, such as RTOS, and timing critical project.

Further effective usage of the tool is detailed in the following section on emulation functions.

4.3.6 Pros and Cons

Pros

- Can work with faulty or incomplete hardware
- Do not use target MCU resource
- Can work standalone without a target
- Needy to solve complex problem
- Provide advance functions:
    - Complex Trace and break control.
    - Performance analysis & Coverage function

Cons

- Costly
- Need in-depth understanding to use it more effectively
- Dedicated user cable connection that may causes intermittent issues

5. Emulation Functions/Features

In this section, various commonly available functions in all ranges of tools are explained to give the reader a better understanding when using these functions. Various dedicated features and functions that are available only in emulator are also highlighted.

5.1 Load

Load of user code to the MCU is a MUST HAVE function. However the intelligence will depend on the PC software used. In most Renesas Tools such as the SLP CPU Board, E7 and the E6000 emulator, a common user interface, HEW is used. This GUI will allow user to download different types of code, such as HEX , Bin, S-Record(mot), and Elf/Dwarf(abs). The Elf/Dwarf file contains specific information for C source level debugging.

Loading of file is only successful if there is a writable memory space available.

Internal ROM memory is loadable because it is implemented with a RAM inside the emulator.

It is impossible to load to:

- ROMless area when there is no external or optional emulation memory available.
- DRAM area when the DRAM is not initialized.

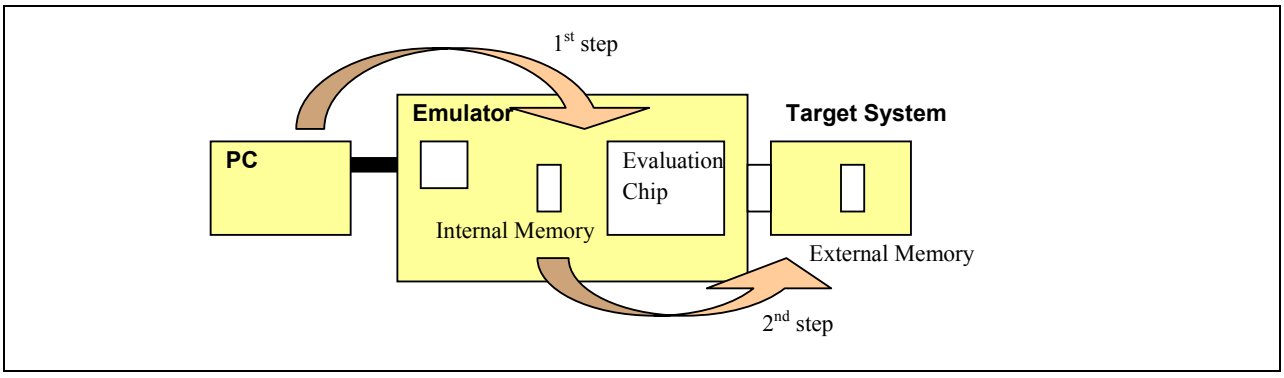
5.2 Memory Access

Memory can be accessed in various forms: Byte, word, and float.

5.3 Download/ Upload Speed

It can be understood that a serial interface emulator will have a slower download speed than a PCI bus interface emulator.

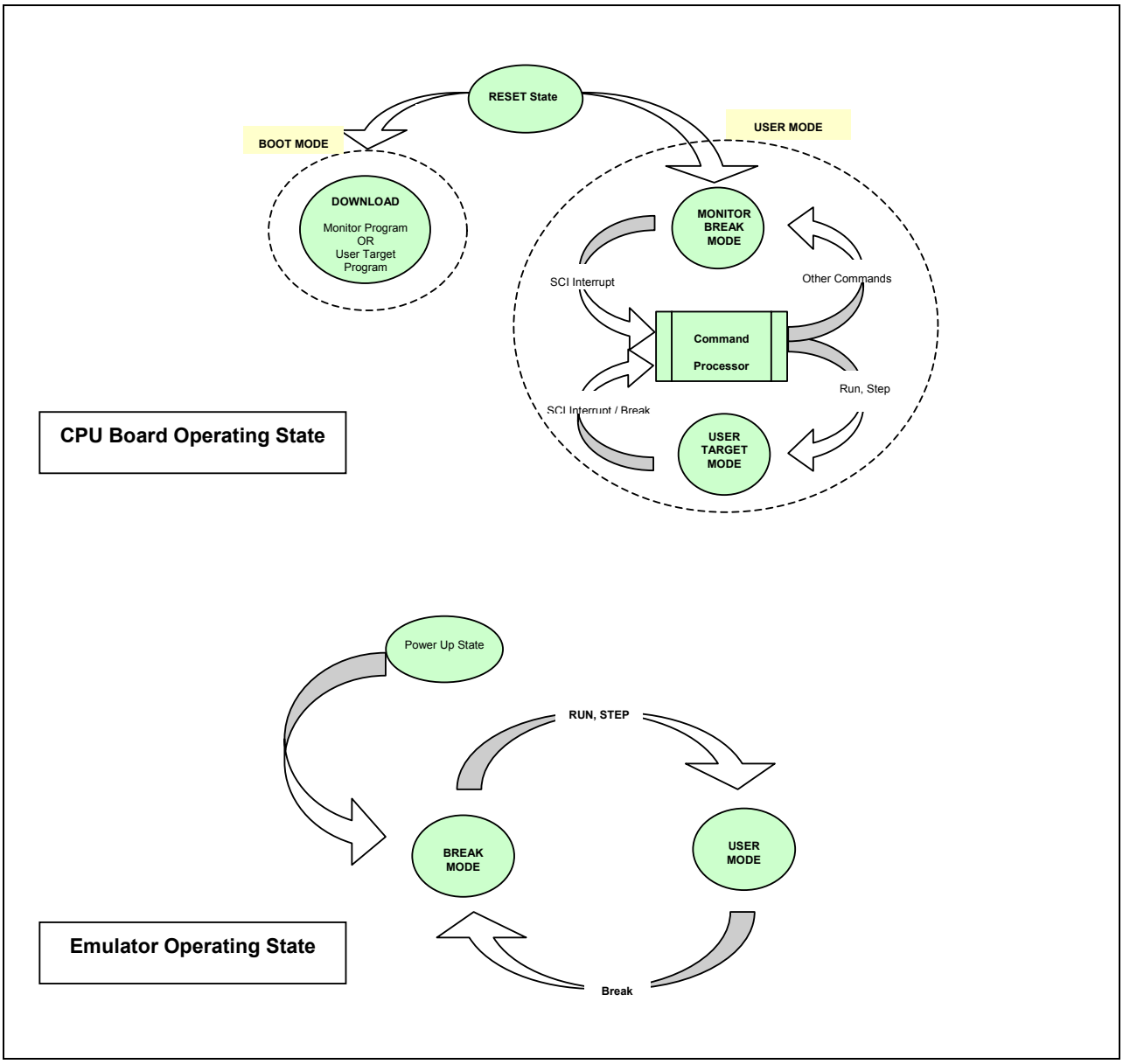
However it may be surprising to know that the downloaded area will also affect the download speed. The following illustration may explain this effect.



If the downloaded area is within emulator (internal memory), the speed will be fast. However if the downloaded area is at external target area, it will need to transfer the data from PC to emulator (internal), and through the evaluation chip, transfer the data out to the external target area. This mechanism will be further slowed down if the user configure the evaluation chip to a slow operating frequency.

5.4 Run

Run is a basic command that must exist in all tools. In “JTAG-like” and Emulator, the target system will be processed (Run) when the command is executed. Otherwise, the MCU will be operating in another mode (generally termed as “Break” mode). In the CPU board case, the MCU start to operate whenever the power is supplied. It will be running in a “command wait” state. When the “run” command is received from the host, it will jump to the user code routine for execution.



5.5 Break

"Break" is defined as a interrupting action that cause an executing code (User mode) to exit to a monitoring or command–wait state (Break mode)

There are two types of break: Hardware & Software.

These are named after their basic implementation.

One distinct characteristic for these two breaks is the PC condition when the break occurred.

For Hardware break, the PC will stop after the event had occurred.

e.g. If condition is (Address=H'1000) → The stopped PC may be H'1002 (depending on MCU)

For Software break, the PC will stop at the condition set.

e.g. If condition is (Address=H'1000) → The stopped PC is H'1000

5.5.1 Hardware Break

Hardware break is implemented by hardware mean. The condition must be matched before a break occurs. The complexity of this break condition enables the capturing of critical condition for evaluation.

Several possibilities of break conditions are:

1. Address or Address range detection
2. Data or Data Masking detection
3. Access type (Read or Write)
4. Access Area (ROM, RAM, DRAM)

A combination of setting is possible, such as "AND", "OR", "IF-ELSE" conditions. In E6000 emulator, the complex events controller allows cascading of events up to six levels. Conditions such as number of occurrences, and number of cycles after condition are matched, are also available.

This will enable the capturing of complex events such as:

- Write to variable COUNT (address = H'1234) with data 10
- Write to data table (address range H'1000 to H'2000) with data H'FF
- Access to variable LOG for 100 times.
- End of ISR (address=H'2024) and break after 20 cycles (To trace the events occur after the interrupt)

5.5.2 Software (PC) Break

The common term used is PC break.

Users can click at the code line to set the breakpoint in HEW. The implementation is a replacement with a dedicated code, causing the MCU to go into another mode. In some CPU board implementation, the TRAP instruction is used.

Since this is a software implementation, this cannot be used at read-only memory, such as Flash, E2PROMand etc. In such condition hardware break is used instead.

The number of Software breaks is normally limited to 255. If users set up 255 Software breakpoints, the PC GUI will need to replace 255 instructions before each execution. This will slow down the reaction time of each “RUN” execution. This delay will be more obvious in CPU board implementation.

Another significant issue will happen for DRAM operation. If user set a Software breakpoint at the DRAM area before the DRAM is being initialized, the PC breakpoint will never be correctly written to the location.

5.6 Emulation Control (Guarded & Write-protected area) / Address Mapping

This is unique in Emulator. Whenever a device is selected, a default mapping will be set. This will define the ROM area as write protected and reserved area as Guarded. By default external memory is also defined as Guarded. User will have to set this external memory area to external read/write if they are to be used (This step is called mapping).

In all MCU, there are 5 main areas and their possible mapping are:

MCU area	Possible Mapping	Remark
On-chip ROM	On chip write - protected	
On-chip RAM	On chip read / write	
IO	On chip read / write	
External area	Guarded External write-protected External read / write	User has an option to change
	Optional Memory write-protected Optional Memory read / write	Explain in next section User can temporary use this area for debugging when target is not ready
Reserved area.	Guarded	

The objective of the “Guarded” & “Write-Protected” control mechanism is to alert the developer that the executing code is having problems, such as program runaway, or performing of write-to ROM area. In some project management, developers may like to guard/reserve certain area so as to prevent individual code from overlapping with other.

5.7 Emulation / Optional Memory

Emulation memory is a general term used to address memory used in the emulator. Users may call the SRAM used to emulate the MCU's on-chip ROM & RAM as emulation memory. However to avoid confusion, it is good to refer the MCU internal memory as on-chip ROM & on-chip RAM.

In emulator, it is common to provide option to "expand " the emulation memory. In this case, this is referred to the extra memory provided in emulator for temporary emulation before target board is ready.

This must be mapped to external area of the MCU memory map. Thus, instead of mapping to external read/write or external write protected, user will have to map it to optional memory read/write or optional memory write-protected.

In the case whereby target is connected, and the emulator mapping is set to optional memory (internal). The emulator will have its access, set to the internal optional memory. Contention will not occur.

5.8 Trace

Trace is an essential function for developers to evaluate the executed codes till last break condition.

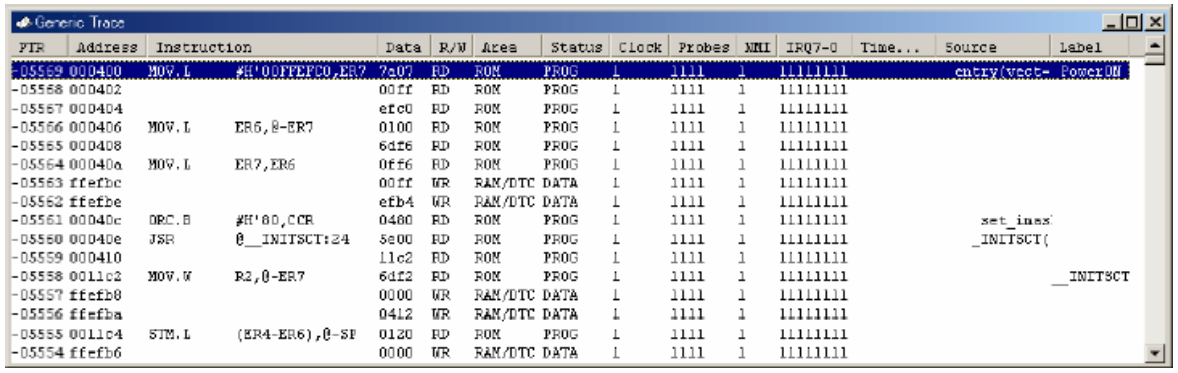
Example of E6000 emulator's trace:

+ Trace Depth: 256K cycles
+ Each cycle contains information of:
  - Address
  - Data
  - Access type (Rd/Wr)
  - Access area (ROM, RAM DRAM…)
  - External Probes
  - Interrupt (IRQ[0-7])
  - Time Stamp

Examples of usage:

- When a write protect break occurs, the trace may tell the developers that their pointer had been over decremented to reach the ROM area when writing data.
- A preset condition is reached whereby the routine reads a data pattern from some external source. The trace may tell the developers that the calculation done is wrong…

Example of E6000 captured trace information



5.8.1 Trace Depth

Technically the trace depth is expected to be as deep as possible, as more information will be available.

However in most cases, only the last few cycles are examined. This is due to the fact that it is the predefine conditions (events break, mapping break…) that had stopped the execution. And the causes of this stoppage can normally be traced in the last few cycles.

The deeper trace will be beneficial in RTOS applications such as analysis of sequence of events.

5.8.2 Trace Filtering

There is always a limit to a trace depth. Thus by setting a filter, developers can capture more information base on the same trace depth. This also allows easy reading of trace information, as developer will be focus on analysis of particular routine, instead of a large junk of data.

5.8.3 Time Stamping

Time stamping is the captured time taken for an instruction. It will be stored accumulating. This will be useful for the measurement of a routine run time, especially when trace filter is activated. Several instructions may be excluded from the trace buffers.

5.9 Single Step (Step In/Out/Over)

Single step provide a control environment for developers to monitor their coding flow under a static condition. The function, Single Step performs execution of instruction at the current program counter. If any interrupt is asserted, the interrupt service routine will not be serviced.

When stepping is performed, Step mode (as in HEW) will determine how each Single Step is performed.

Step Mode consists of three modes:

- **Auto** The execution mode will depend on the active window. i.e. when step instruction is activated in a C Source window, a C-source level step will be invoked.
- **Source** When Step instruction is executed, user will see a C-source level step (if available). i.e. a series of assembly code is run in the background.
- **Assembly**: When step is executed, the current assembly code located at current PC will be executed. If the current window is a C source window, the disassembly window will pop up.

There are generally three common terms used for Single Step:

- **Step-In** executes a single instruction only. In simple words, it will step into the next C or assembly instruction (depending on step mode selection)
- **Step-Out** executes till it has branched out of the current routine.
- **Step-Over** executes a function call (and any function call called by the function) and halt at the next instruction.

During the “single stepping” of C instruction, different emulators may perform it differently. It may single step through all the assembly codes that interpret the C instruction, or it may “run” through the assembly codes that interpret the C instruction. Technically both methods are acceptable as these provide only a static mode of testing.

Single step implementation in CPU board is rather complicated than the other tools. It is a simulated step in the CPU board. Thus other complication will arrived if there are interrupting events.

5.10 Intrusiveness / Real Time

All tools are designed to be real time. However in circumstances whereby developers want to obtain information of the running MCU, certain degree of intrusive probing will be required. It is technically impossible to access MCU information (such as I/O registers, target memory data…) without stopping the MCU normal operation.

The MCU is switched from the "user" mode to "break" mode, to perform the access (read/write the register/memory), and switched back to "user " mode in the shortest possible time. This is frequently referred as "Parallel On the Fly"(POTF), which will be elaborated in the following section.

If users do not activate this feature, the MCU will be free to run at its own speed. POTF can be disabled in emulator system setup.

5.11 Monitor/Control - Parallel On the Fly and Bus Monitor

During the development, developers may wish to access the variable (memory) values during the program execution.

The two implementations and respective functions are:

1. Parallel On The Fly (POTF) – allow intrusive read and write of memory content
2. Bus monitor – allow non-intrusive read of memory content

POTF – When HEW memory window is refreshed, the MCU will be interrupted to update the latest data. If the memory window content is altered, the MCU will be interrupted to change the memory content.

Bus Monitor - The address and data bus is being monitored with changes. Once there is any activity (read/write) the Bus Monitor data window will be updated. The monitoring nature guaranteed zero interruption to the running MCU.

The intrusive nature of POTF may be a negative factor for some applications. In communication application, the interruption may cause lost of communication data. However, if developer knows how to manipulate the function, it will be a very effective function.

5.12 External probe

External Probe is used in emulator to detect external events. These probes can be used to tap onto target board circuitry. When an event (such as Probe 1 is Low) occurs, a break can be triggered. The conditions of the available probes can be configured as low, high or edge trigger. This will enable better tracking of program flow when external events occur.

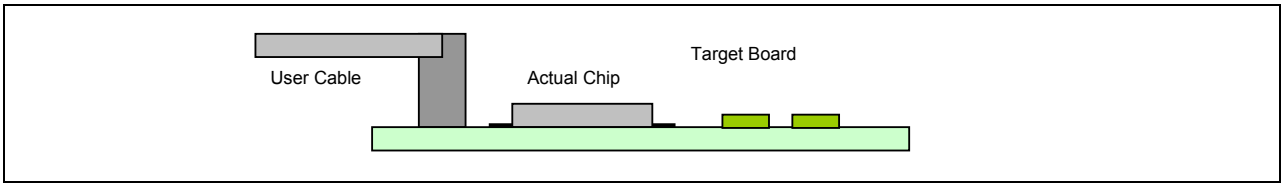
Since the probes' logic level is captured into the trace memory, this feature can be used as a Logic Analyzer. When the probes are tapped to different points of the target, developers can analyze their running code against the events captured by the probes.

5.13 Target connection

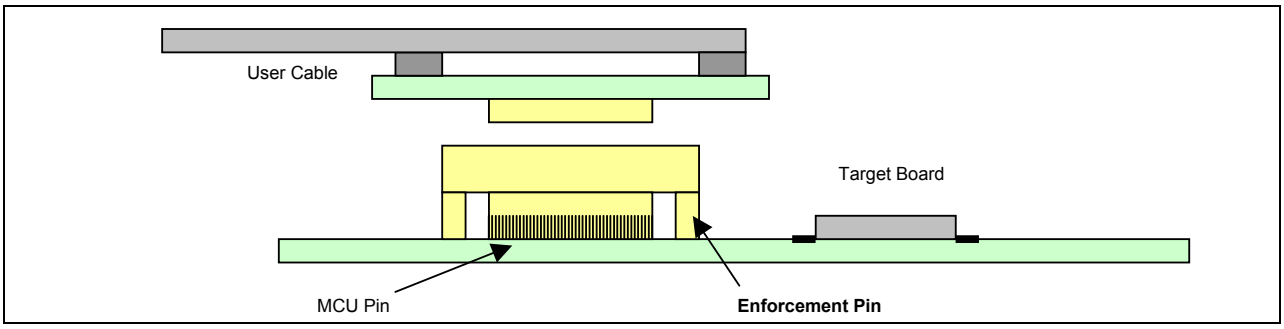
The following illustrated both the “JTAG-like” and “Emulator” connection.

Target connection for “JTAG-like“ emulator is simply an n-wire connection to the MCU. The connection is reliable.

Example of “JTAG-like” connection:



Emulator connection to a target system is via a dedicated user cable

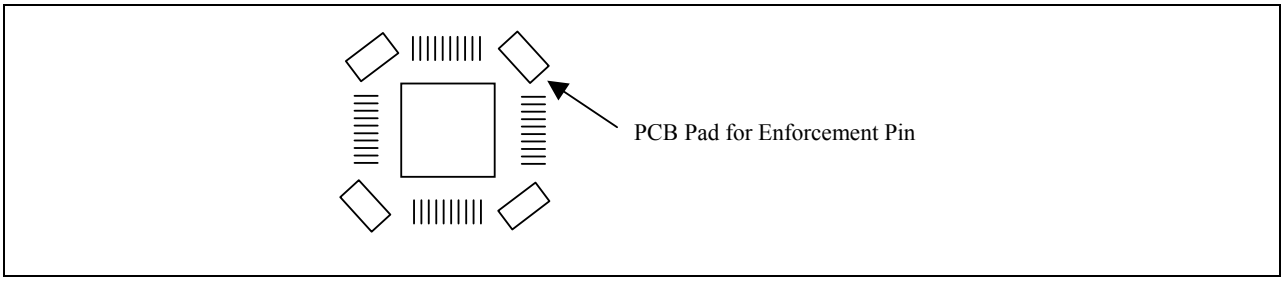


The target connection issues for emulator can be classified as

1. Intermittent connection
2. Delay & Noise margin
3. Signal control (NMI, Reset, Mode, Standby, Clock)
4. Power Supply - Voltage follower
5. Oscillation Clock
6. Pull-up resistor and Cabling

5.13.1 Intermittent Connection

Developers have to pay extra care when dealing with the target connection. PCB design for the target connector is also very important. This forms the basic strong foundation for the connector. It is strongly advise that PCB pad is designed for the enforcement pin. The soldering of this pin to the pad will provide a much better grip. Extra care must be catered especially for some big and bulky user cable.



5.13.2 Delay and Noise Margins

Although every effort is made to minimize the delay and noise issues in the emulator design, these are still inevitable. Developers are advised to put this into consideration when debugging their target system. In general, the cable delay from the emulator to the target is estimated to be 4ns. If any interface circuitry is introduced in between, an additional of 4 ns will be introduced (Depending on device used).

5.13.3 Signal Control (NMI, Reset, Mode, Standby, Clock)

Unlike other ranges of tool, Emulator is designed to have full control of the target system. The target system signal such as NMI, reset, mode pins, standby, and clock are controlled.

When a target system is not available, emulator is still capable to control the emulation based on development needs.

Example: - Setting of operating mode to be extended mode and the running frequency to be 14MHz.

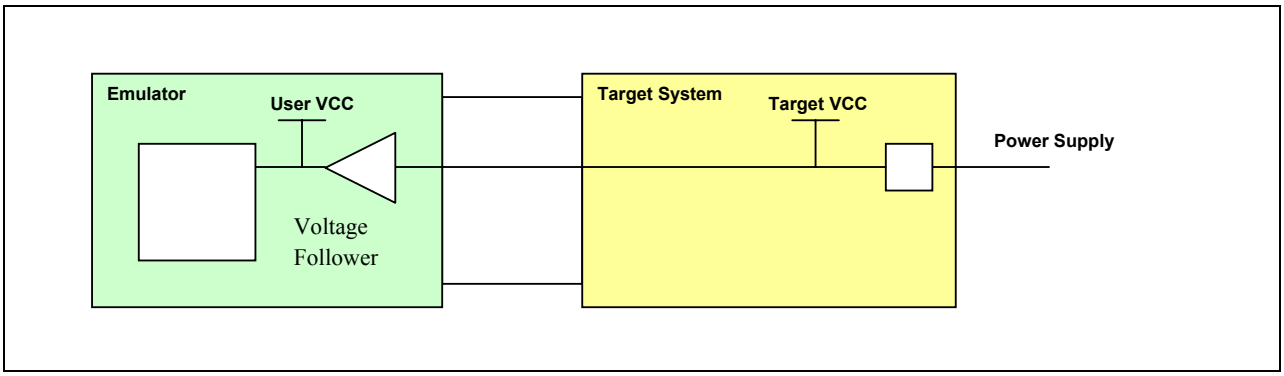
When a target system is connected, user may set the setting to be based on target setting instead of emulator setting. However proper emulation may not be guaranteed when target is connected. In this case, user may like to make adjustment to troubleshoot their system.

Examples: - Change to emulator internal lower operating frequency and observe system response

- Mask off external target watchdog reset or exception NMI, in order to have a controlled emulation.

5.13.4 Power Supply - Voltage follower

When a target is connected to an emulator, developers may have the worry of emulator loading of the target supply.

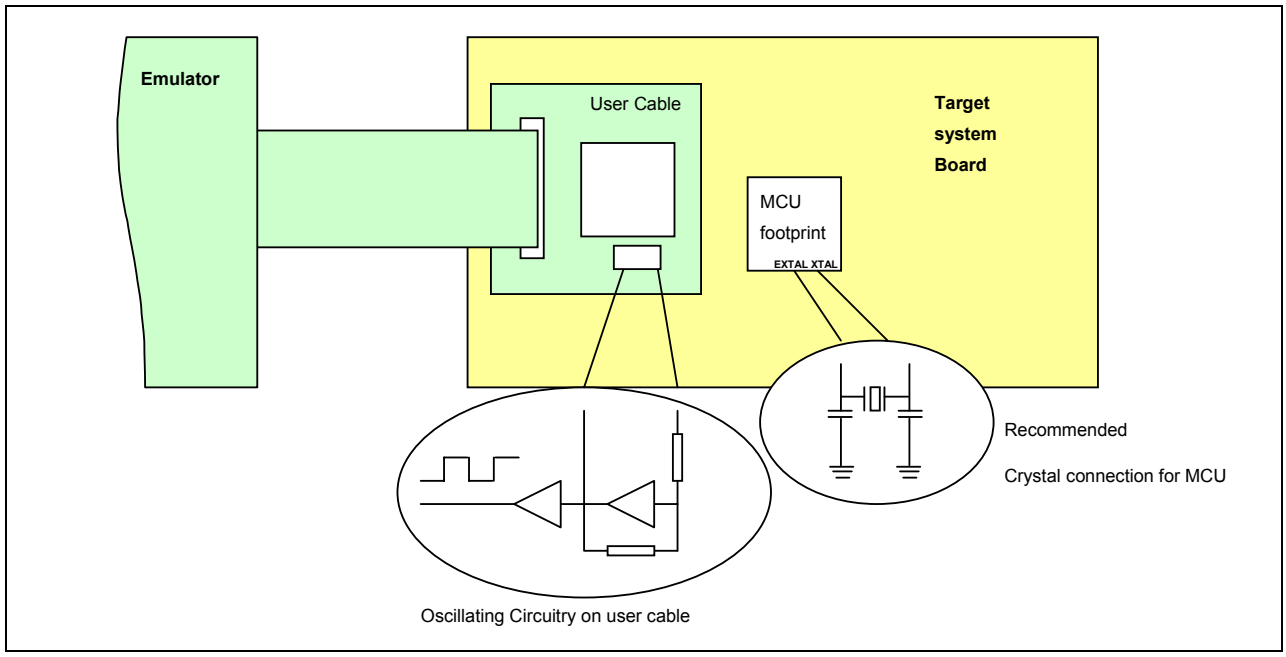


From the above diagram, Hardware developers can rest assured that the emulators will not draw current from the target. The emulator will generate an internal voltage (User Vcc) for internal operation, which is based on a target system supply (Target Vcc). When current measurement is made on the target, developers will have to take into the account that the emulating MCU is not consuming the target power.

5.13.5 Oscillation Clock

It is a common design to place a crystal and two capacitors at the MCU pins; EXTAL and XTAL. Oscillation is possible as there is a built-in circuitry to cause crystal to go into resonance. In an emulator, the emulating MCU (evaluation chip) may be place in the main unit or at the target user cable header. If the evaluation chip is placed in the main unit, oscillation is impossible. Thus, such emulator will have their oscillating element placed at the target user cable (see illustration). In this case, an oscillating clock will be fed through the long user cable to the emulator main unit.

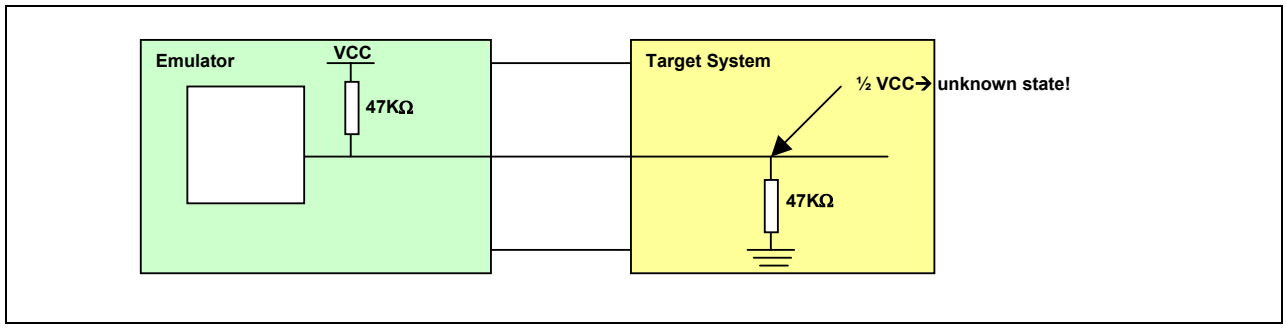
In some cases, users may wish to connect the target to the emulator directly (This is to avoid the weaker link of the actual footprint user cable). Users have to understand that their clock input must be oscillating. Alternatively user can use the emulator internal clock selection.



5.13.6 Pull-Up resistor

Within the emulator, it is common to have pull up resistors at most input and output pins. This is to ensure known state and improve driving current from the long cabling to target system. For analog pins (such as ADC & DAC) pins, there are no pull up resistors. Two main issues may arise:

- When actual chip replaced the emulator operation, developer forgoes the resistor effect.
- During emulation, developer places a similar value pull-down resistor. This will cause a voltage divider effect causing a unknown state feeding into the pin.



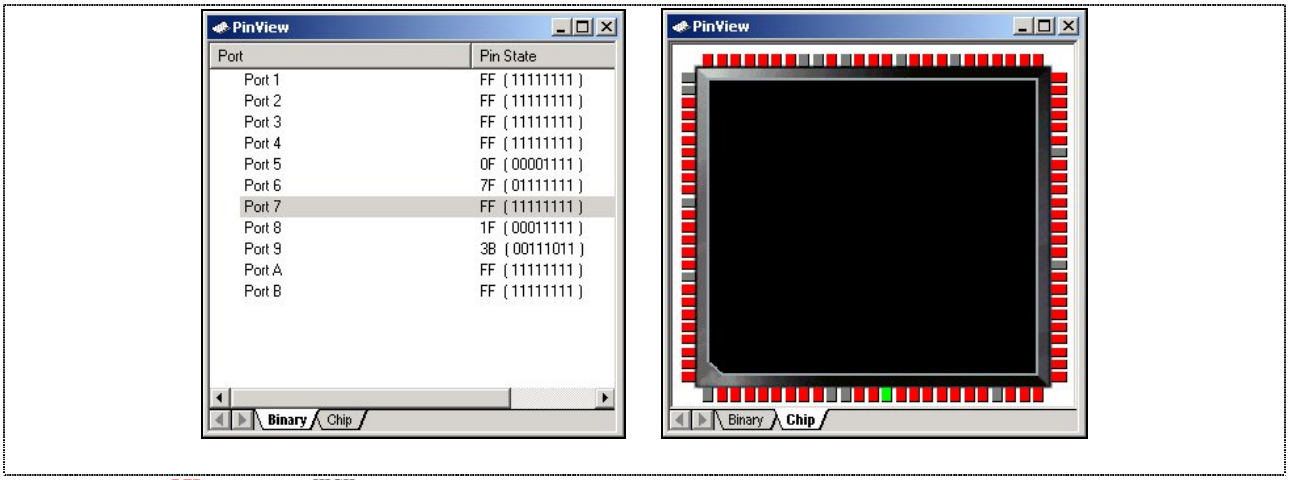
Users are advised to refer to emulator detailed interface description available in each user manual.

In some cases, emulator allow user to remove the pull up resistors by merely unplugging it from the DIL sockets.

5.14 Pinview

Connectivity between emulator and target system is always a concern. In Renesas CE emulator series, Pinview is implemented to address this problem. As the name implied, it is used provide another mean to monitor the Pin state.

Example: A snapshot of CE300H Pinview window



The emulator constantly monitor all the pins interface at the target connection through hardware means. As such this will not cycle steal / intrude the emulating MCU operation. This information is fed back to the PC GUI (HEW) to be displayed in a graphical mean. This will provide developer with an instant view of all Pins status. (High or Low). This will provide some basic mean of gauging the running program (example BUSY signal is always asserted…) At static mode (when program is not running), this will give a better judgment for the developer, as each MCU pins status can be visually accessed instead of having a multi-meter to probe all the pins.

5.15 MCU and Emulator Differences

The “power up” & “reset” state of the MCU and emulator are different. Generally the start up state of the MCU’s registers is unknown, whereas the emulator will initialize the registers to a known value. However this may not be important for developer if HEW is used for a C project. (The HEW project generation will setup all registers accordingly)

Example: Differences Between H8 MCU and Emulator

Status	Register	Emulator	H8 MCU
Emulator Initialization (Power on)	PC	Power on reset vector value	Power on reset vector value
	ER0 to ER6	H’00000000	Undefined
	ER7 (SP)	H’0FFF10	Undefined
	CCR	The I mask is set to 1 and the other bits are undefined (B’1 xxx xxxx)	The I mask is set to 1 and the other bits are undefined (B’1xxx xxxx)
Emulator Initialization (Reset CPU command)	PC	Power on reset vector value	Power on reset vector value
	ER0 to ER6	Undefined	Undefined
	ER7 (SP)	Undefined	Undefined
	CCR	The I mask is set to 1 and the other bits are undefined (B’1 xxx xxxx)	The I mask is set to 1 and the other bits are undefined (B’1 xxx xxxx)

A general difference of the User Interface:

- The emulator’s user system interface is provided with pull-up resistors and/or a buffer, causing the signals to be delayed slightly.
- The pull-up resistors will change the high-impedance signals to high-level signals.
- The resolution of the Analog to Digital conversion will have a slight degradation.
- Load capacitance of the emulator as compared to the actual chip will be larger.
- Crystal oscillator can only used if the actual footprint user cable is used (there is an oscillating circuitry built on-board). If user uses the direct method of connecting to the target system, the actual clock signal has to be connected to the EXTAL pin (**NOTE**: there is no XTAL pin).

6. Summary

There are a variety of tools in the market, which cater to different needs and requirements. The most important factor to the users, is the understanding of the tool. Without a good understanding, an expensive and sophisticated tool may even cause more confusion to the development work than any other tools.

This application note has highlighted the various characteristics of different tools. Users have to apply this know-how to select the tool for their development. The effective usage of the tool will enable a more efficient development cycle.

Reference

www.embedded.com

The Practice of Programming by Brain W.Kernighan & Rob Pike (Addison –Wesley)

Fundamentals of Embedded Software where C and Assembly Meet by Daniel W.Lewis (Prentice Hall)

Programming Embedded Systems in C and C++ by Michael Barr (O'REILLY)

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	March 04	—	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.