Integrated Development Environment for RX Family

Migration to New Integrated Development Environment “CubeSuite+”: Emulator

Renesas Electronics Corporation
MCU Business Unit
MCU Software Division
MCU Tool Product Marketing Department

February 13, 2012 Revision 1.01
Introduction

This document describes how to migrate the High-performance Embedded Workshop for RX Family to CubeSuite+ and how to operate E1 and E20 emulators in the CubeSuite+ environment, on the basis of CubeSuite+ V1.01.00.

For toolchains, refer to *Integrated Development Environment for RX Family Migration to Integrated Development Environment “CubeSuite+”: Build*. Also refer to the tutorial guide provided by CubeSuite+ for how to use tools. The tutorial guide is available by selecting [Help] -> [Tutorial] from the CubeSuite+ menu.
Contents

1. Opening a Workspace of High-performance Embedded Workshop by using CubeSuite+
2. Changing the Debugger (E1/E20 Emulator or Simulator)
3. Changing the MCU
4. Where Do We Make Settings when Connecting an Emulator?
5. Connecting an Emulator
6. Disconnecting the Emulator
7. Starting an Emulator with Hot Plug-in
8. Entering an ID Code
9. Downloading a Program
10. Registering Additional Download Files
11. Starting/Stopping a Program
12. Setting the Start/Stop Function
13. Setting Trace Acquisition Conditions
14. Setting Trace Start/Stop Conditions
15. Setting Trace Extraction Conditions
16. Viewing/Changing Memory Data and Variables While the Program is Running
Contents

17. Automatically Updating Memory Data and Variables While the Program is Running
18. Setting Breakpoints
19. Causing a Break on Access to a Variable
20. Using the Performance-Measurement (Timer) Function
21. Filling Memory
22. Saving Memory Data
23. RAM Monitoring (When the E20 is Used)
1. Opening a Workspace of High-performance Embedded Workshop by CubeSuite+

The CubeSuite+ can open a workspace created by the High-performance Embedded Workshop. Perform the following procedure to open a workspace.

(1) Click the [Start] button at the upper left of the toolbar to open the start panel.

[Start] button
1. Opening Workspace of High-performance Embedded Workshop by Using CubeSuite+ 

(2) Click the [GO] button to the left of [Open Existing CubeSuite/High-performance Embedded Workshop/PM+ Project].

(3) The [Open Project] dialog box opens. Select [Workspace File for HEW(*.hws)] for file type, specify a workspace created by the HEW, and click the [Open] button.
1. Opening Workspace of High-performance Embedded Workshop by Using CubeSuite+

(4) If the HEW workspace contains multiple sessions, the [Select Session] dialog box opens. Select the session you want to use and click the [OK] button.

(5) Then the [Select Device] dialog box opens. Select the name of the device you want to use and click the [OK] button. The HEW workspace is converted to a CubeSuite+ project.

Caution

Once a device has been set, it can no longer be changed. For how to change the device, see “3. Changing the MCU.”
2. Changing the Debugger (E1/E20 Emulator or Simulator)

The HEW allowed users to select a debugger (E1/E20 emulator or simulator) in the process of changing the debug session or the target shown in the [Debug Settings] dialog box.

The CubeSuite+, on the other hand, allows users to select a debugger on the project tree. The procedure to change the debugger is described on the following pages.
2. Changing the Debugger (E1/E20 Emulator or Simulator)

(1) The debug tool name (debug tool) on the project tree panel indicates the currently selected debugger. The following example shows that the RX simulator is selected:
2. Changing the Debugger (E1/E20 Emulator or Simulator)

(2) To change the debugger, right-click the debug tool name (debug tool) to open a pop-up menu. Select [Using Debug Tool] from the pop-up menu to select the debug tool you want to use.

Right-click
Select the debug tool you want to use.
3. Changing the MCU

The CubeSuite+ does not allow changing of the MCU for a project that has already been created.

To change the MCU, a new project must be created based on an existing project. Perform the following procedure to create a project:

(1) Click the [Start] button at the upper left of the toolbar to open the start panel.
(2) Click the [GO] button on the left of the [Create New Project] field.
3. Changing the MCU

(3) The [Create Project] dialog box opens. Select the name of the MCU you want to use in [Using microcontroller].

(4) Select the [Pass the file composition of an existing project to the new project] checkbox, select the original project, and then select the [Copy composition files in the diverted project folder to a new project folder] checkbox.
3. Changing the MCU

(5) Specify [Kind of project] and [Project name] and click the [Create] button. The newly created project is based on the original project but the MCU has been changed.

Select [Empty Application (CC-RX)].

Enter project name.

Click [Create] button.
4. Where Do We Make Settings when Connecting an Emulator?

In the HEW, the [Initial Settings] and [Configuration Properties] dialog boxes open to make settings when connecting an emulator. In the CubeSuite+, on the other hand, you need to make settings on the [Properties] panel before connecting an emulator by taking the following procedure.

Double-click the debug tool name (debug tool) on the [Project Tree] panel to open the Properties window of the debug tool.
4. Where Do We Make Settings when Connecting an Emulator?

The following figures show the correspondence between settings on the HEW’s [Initial Settings] and [Configuration Properties] dialog boxes and settings on the [Properties] panel of the CubeSuite+.

(1) [Device] page of the [Initial Settings] dialog box in the HEW

Note: It is not possible to change the device on the CubeSuite+.
For how to change the device, see “3. Changing the MCU.”
4. Where Do We Make Settings when Connecting an Emulator?

(2) [Startup and Communication] page of the [Initial Settings] dialog box in the HEW.
4. Where Do We Make Settings when Connecting an Emulator?

(3) [MCU] page of the [Configuration Properties] dialog box in the HEW
4. Where Do We Make Settings when Connecting an Emulator?

(4) [System] page of the [Configuration Properties] dialog box in the HEW

![Configuration Properties Dialog Box]

- Switching function
  - [ ] Trace conditions
  - [ ] Realtime EAM monitor

- Debug function
  - [ ] Debugging the program re-writing the internal PROGRAM ROM
  - [ ] Debugging the program re-writing the internal DATA FLASH

- System
  - Debug the program re-writing the on-chip PROGRAM ROM: Yes
  - Debug the program re-writing the on-chip DATA FLASH: Yes
  - Execute the specified routine immediately before execution of the user program: No
  - Execute the specified routine immediately after the user program: No

- Trace
  - Usage of trace function: Trace
  - Operation after trace memory is full: Overwrite trace memory and continue execution
  - Trace data type: Branch

- Timer
  - Use 64bit counter: No
  - Operating frequency (MHz):  

- Debug the program re-writing the on-chip PROGRAM ROM
  - Select whether or not to debug the program which rewrites the on-chip program ROM.
4. Where Do We Make Settings when Connecting an Emulator?

(5) [Internal flash memory overwrite] page of the [Configuration Properties] dialog box in the HEW

Note: The erasing and overwriting settings are made for the entire flash area, and cannot be made for each block.
4. Where Do We Make Settings when Connecting an Emulator?

(6) [External flash memory] page of the [Configuration Properties] dialog box in the HEW

Note: USD files used in the HEW can be registered as is. They do not need to be modified even if a pre-download execution script and a post-download execution script have been registered.
5. Connecting an Emulator

Select [Debug] -> [Connect to Debug Tool] from the CubeSuite+ menu to establish connection to the selected emulator (debug tool). Upon completion of the connection, the debug tool name appears on the status bar at the bottom right of the window.

Note: If an ID code has been written in the MCU, set an ID code in advance according to “8. Entering an ID Code.”
6. Disconnecting the Emulator

To disconnect the emulator, select [Disconnect from Debug Tool] from the menu or click the \( \times \) button on the debug toolbar.
7. Starting an Emulator with Hot Plug-In

The following describes a procedure to connect an emulator by using the hot plug-in facility.

(1) Select [Debug] -> [Hot Plug-in] from the CubeSuite+ menu.

(2) The following message appears. Connect the emulator to the target board and then click the [OK] button to start up the emulator.

Note: If an ID code has been written in the MCU, set an ID code in advance according to “8. Entering an ID Code.”
8. Entering an ID Code

In the HEW, the [ID Code verification] dialog box opens at startup if an ID code has been written in the MCU. In CubeSuite+, on the other hand, an ID code must be set on the [Properties] panel before the emulator is started up. Set an ID code by referring to the following figures:
9. Downloading a Program

Selecting [Debug] -> [Download] from the menu or clicking the button on the debug toolbar starts downloading specified files.

Selecting [Debug] -> [Build & Download] from the menu or clicking the button on the debug toolbar builds a project and then starts downloading the specified files.

If no debug tool is connected, connection to a debug tool must be established before downloading.
10. Registering Additional Download Files

(1) Select [Download files] and click [...] on the right.
(2) The [Download Files] dialog box opens. Click the [Add] button.
10. Registering Additional Download Files

(3) Specify the file name and file type in the [Download file information] field and then click the [OK] button.

Note: When downloading is performed, all of the registered files are downloaded. To download only desired files, set [Download object] and [Download symbol information] to “Yes” in this window only for files you want to download.
11. Starting/Stopping a Program

You can start or stop a program and reset the CPU from the menu or toolbar in the same way as the HEW (see below).
12. Setting the Start/Stop Function

You can set the Start/Stop function in the [Debug Tool Settings] sheet on the [Properties] panel (see below).
13. Setting Trace Acquisition Conditions

You can set trace acquisition conditions in the [Debug Tool Settings] sheet on the [Properties] panel (see below).

Note: CubeSuite+ V1.01.00 does not support advanced settings (including timestamps). The next and subsequent versions will support advanced settings.
14. Setting Trace Start/Stop Conditions

14.1 You can use the editor panel to specify an address where trace acquisition will start or stop (see below).
14. Setting Trace Start/Stop Conditions

(1) Right-click the event area of the line at which you want to start or stop tracing. This opens a pop-up menu.

(2) Select [Set Trace Start Event] or [Set Trace End Event].

(3) is displayed in the line for which [Set Trace Start Event] has been set, and is displayed in the line for which [Set Trace Stop Event] has been set.
14. Setting Trace Start/Stop Conditions

14.2 You can use the watch or editor panel to start or stop tracing on data access.
(1) On the watch or editor panel, right-click the variable that you want to start or stop tracing when it is accessed. This opens a pop-up menu.
(2) Select [Trace Output] (or [Trace Settings] on the editor panel) and select [Record Start R/W Value] or [Record End R/W Value].
(3) Enter a value to set a data condition.
14. Setting Trace Start/Stop Conditions

All conditions that have been set can be checked on the event panel.
15. Setting Trace Extraction Conditions

You can set trace extraction conditions on the watch or editor panel (see below).
15. Setting Trace Extraction Conditions

(1) On the watch or editor panel, right-click the variable that you want to use as a trace extraction condition. This opens a pop-up menu.
(2) Select [Trace Output] (or [Trace Settings] on the editor panel) and select a trace extraction condition.

Right-click here.

Select [Trace Output] or [Trace Settings].

Select a trace extraction condition.
16. Viewing/Changing Memory Data and Variables While the Program Is Running

To view or change memory data and variables while the program is running in CubeSuite+, make settings on the [Properties] panel by using the following procedure:

(1) Open the [Debug Tool Settings] sheet on the [Properties] panel of the debug tool.
(2) Set [Access by stopping execution] in the [Access Memory While Running] field to [Yes]. Memory data and variables can be viewed while the program is running.

If [No] is selected, “***” is displayed on the memory panel while the program is running.
17. Automatically Updating Memory Data and Variables While the Program Is Running

To automatically update memory data and variables via CubeSuite+, make settings on the [Properties] panel by using the following procedure:

2. Set [Access by stopping execution] and [Update the display during execution] in the [Access Memory While Running] field to [Yes].

Information displayed on the memory and watch panels is automatically updated while the program is running.

To change the update interval, modify the [Update interval] value.

- Change two items to [Yes].
- Set update interval.
- Variables registered in the watch panel and memory data in the memory panel are automatically updated while the program is running.
18. Setting Breakpoints

(1) You can set breakpoints in the main area (enclosed by a red line in the figure below) on the editor panel of CubeSuite+.
Single-clicking a line with an address sets a breakpoint. Single-clicking a line for which a breakpoint has been set deletes the breakpoint.
18. Setting Breakpoints

(2) Select a breakpoint type (software break or hardware break) for [Type of breakpoints to be preferentially used] in the [Debug Tool Settings] sheet on the [Properties] panel. (Hardware break is selected in the example below.)

(3) If the number of breakpoints of the selected type exceeds the limit, the other type of breakpoints are used.

Event marks indicate the types of breakpoints.
- : Software break
- : Hardware break
18. Setting Breakpoints

(4) You can check the breakpoint setting on the [Event] panel. Select [View] -> [Event] from the CubeSuite+ menu to open the [Event] panel. Unnecessary breakpoints can be deleted or disabled on the [Event] panel.
19. Causing a Break on Access to a Variable

You can use the watch or editor panel to make a setting to cause a break on access to a specific variable.

(1) On the watch or editor panel, right-click the variable that you want to cause a break when it is accessed. This opens a pop-up menu.

(2) Select [Access Break] (or [Break Settings] on the editor panel) and select [Set Read Combination Break to], [Set Write Combination Break to], or [Set R/W Combination Break to].

Right-click the variable to open a pop-up menu.

Select [Access Break].

Select a break condition (read, write, or read/write).
19. Causing a Break on Access to a Variable

(3) Enter a value to set a data condition (or leave the box blank if no data condition is needed).

Note: Enter a decimal number here. When entering a hexadecimal number, add “0x” to the head (e.g. 0xAA).
19. Causing a Break on Access to a Variable

You can also specify masking of data conditions in the [Detailed Settings of Access Events] dialog box.

(4) Select [View] -> [Event] from the menu to open the event panel. Click the “+” mark to the left of [Combination Break] for expansion.
(5) Select and right-click the event you want to modify in the [Detail] field to open a pop-up menu. Select [Edit Condition] from the pop-up menu.
19. Causing a Break on Access to a Variable

(6) The [Detailed Settings of Access Events] dialog box opens. Set [Specify the data mask] in the [Data Condition] field to [Yes].

(7) Setting of a mask value is enabled. Enter a mask value. The bits of the data value for which a mask value “0” has been specified is treated as “Don’t Care”.

This dialog box allows settings for masking address conditions, specifying a range, access type, access size, comparison data, and pass count for data conditions in addition to masking data conditions.
20. Using the Performance-Measurement (Timer) Function

While the HEW can measure program execution time with the performance-measurement function, the CubeSuite+ allows measurement of program execution time by the timer. Measurement results can be viewed on the [Event] panel (opened by selecting [View] -> [Event] from the menu).
20. Using the Performance-Measurement (Timer) Function

20.1 You can set measurement start/stop conditions on the editor panel. The procedure is described on the next page.

Note: Start/stop conditions cannot be registered through function names in CubeSuite+.
20. Using the Performance-Measurement (Timer) Function

Set measurement start/stop conditions by using the following procedure.
(1) On the [Editor] panel, right-click the event area of the line at which you want to start or stop measurement. This opens a pop-up menu.
(2) Select [Set Timer Start Event] or [Set Timer End Event].
(3) If two channels are provided, select a timer (Timer 1 or Timer 2) to be used for measurement.

Right-click here.

Select timer start or stop.

Select Timer 1 or Timer 2.
20. Using the Performance-Measurement (Timer) Function

(4) Upon completion of the setting, \( \text{[Set Timer Start Event]} \) appears in the line for which [Set Timer Start Event] has been set, and \( \text{[Set Timer End Event]} \) appears in the line for which [Set Timer End Event] has been set.
20. Using the Performance-Measurement (Timer) Function

20.2 You can set [Measurement item] and [Execution only once] on the [Detailed Settings of Timer Measurement] dialog box.


Right-click here to open a pop-up menu.

Select [Edit Condition].

Set conditions.
20. Using the Performance-Measurement (Timer) Function

20.3 You can set [Display the cycle as time span] and [Use 64 bit counter] in the [Debug Tool Settings] sheet on the [Properties] panel.
20. Using the Performance-Measurement (Timer) Function

20.4 When all settings have been made, execute the program. The time measurement result is displayed at a break.
21. Filling Memory

Memory can be filled (batch change) via the [Memory Initialize] dialog box.
(1) Right-click on the [Memory] panel to open a pop-up menu, and select [Fill] from the pop-up menu.
(2) The [Memory Initialize] dialog box opens. Enter addresses (start address and end address) you want to initialize and initialization data, and then click the [OK] button.

Note: Enter decimal numbers here. When entering hexadecimal numbers, add “0x” to the head of each number.
22. Saving Memory Data

[Data Save] dialog box is used to save memory data via CubeSuite+. Select [Debug] -> [Upload...] from the menu. The [Data Save] dialog box opens. Specify the file name, type, and range of memory data you want to save, and then click the [Save] button.

Note: Enter decimal numbers here. When entering hexadecimal numbers, add “0x” to the head of each number.
23. RAM Monitoring (When the E20 is Used)

CubeSuite+ provides the RRM (Real-time RAM Monitor) function that is equivalent to the HEW’s RAM monitoring. The following describes how to use the RRM function:

23.1 Set [Usage of trace function] in the [Debug Tool Settings] sheet on the [Properties] panel of the debug tool to [Real-time RAM monitor].

Set Usage of trace function to [Real-time RAM monitor].
23. RAM Monitoring (When the E20 is Used)

23.2 In the [Debug Tool Settings] sheet, set [Access by stopping execution] to [No], [Update the display during execution] to [Yes], and [Enable the automatic update of realtime display] to [Yes].
Set a desired update interval for [Update interval].

Make the following settings:
[Access by stopping execution]: No
[Update the display during execution]: Yes
[Update interval]: Desired update interval
[Enable the automatic update of realtime display]: Yes

Note: If [Access by stopping execution] is set to [Yes], the RRM function is not usable.
23. RAM Monitoring (When the E20 is Used)

23.3 Open the area you want to view with real-time RAM monitoring on the [Memory] panel, and then execute the program. Memory data is displayed by the RRM function.

Example of display by the RRM function

- **00**: Read/fetch
- **00**: Write
- **00**: Read/write
- **00**: No access
- **00**: Loss of data
- **00**: An area other than the real-time display update area was specified while the program is running or acquisition of memory data failed
23. RAM Monitoring (When the E20 is Used)

23.4 Troubleshooting
(1) Values on the [Memory] panel are updated, but the background color remains white.

The RRM function is disabled.
If [Access by stopping execution] in the [Debug Tool Settings] sheet on the [Properties] panel is set to [Yes], the RRM function is not usable.
Set [Access by stopping execution] to [No].
23. RAM Monitoring (When the E20 is Used)

23.4 Troubleshooting

(2) “**” is displayed on the [Memory] panel.

The displayed area is not covered by the RRM function.
The RRM function covers an area of up to 4096 bytes (1024 bytes × 4) that is automatically allocated according to the data on the open [Watch] and [Memory] panels.
If the amount of data displayed on the [Memory] panel is larger than the area allocated for the RRM function, the RRM function is not usable.
In that case, perform the following:
(a) Reduce the number of variables registered in the [Watch] panel or close the [Watch] panel.
(b) Open only the [Memory] panel on which you want to use the RRM function.