V850E2/ML4

Performance Evaluation Software

R01AN1228EJ0100
Rev.1.00
Aug. 24, 2012

Abstract

This document describes a sample code that uses timer array unit A (TAUA) of the V850E2/ML4 to evaluate performance of the user-created tasks (functions).

Products

V850E2/ML4

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

In this application note, the V850E2/ML4 evaluates performance of the user-created tasks (functions). This sample code is designed to have the user-created tasks embedded in it to select and activate the user-created tasks from a serial terminal. It evaluates the performance by counting the number of cycles taken from start to end of the user-created tasks.

As shown in Figure 1.2 "Specification Diagram", the V850E2/ML4 CPU board waits a command which specifies the processing to be executed from several evaluation processing (evaluation command). When the evaluation command is transmitted, it executes a corresponding evaluation processing and transmits the result via serial output.

When a performance evaluation is executed, an evaluation command is transferred to the V850E2/ML4 CPU board from the host PC via the serial cable. For this reason, the host PC is required to install a serial cable and a serial communication application soft such as hyper terminal other than the V850E2/ML4 CPU board. Refer to the section 5.1 "Performance Evaluation" for details.

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows the System Configuration Diagram.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
Timer array unit A (TAUA)	Measures the time for task processing
Asynchronous serial interface (UARTJ)	Receives an evaluation command from the host PC and transmits the evaluation result
Interrupt function	Used for UARTJ receive interrupt and TAUA interrupt

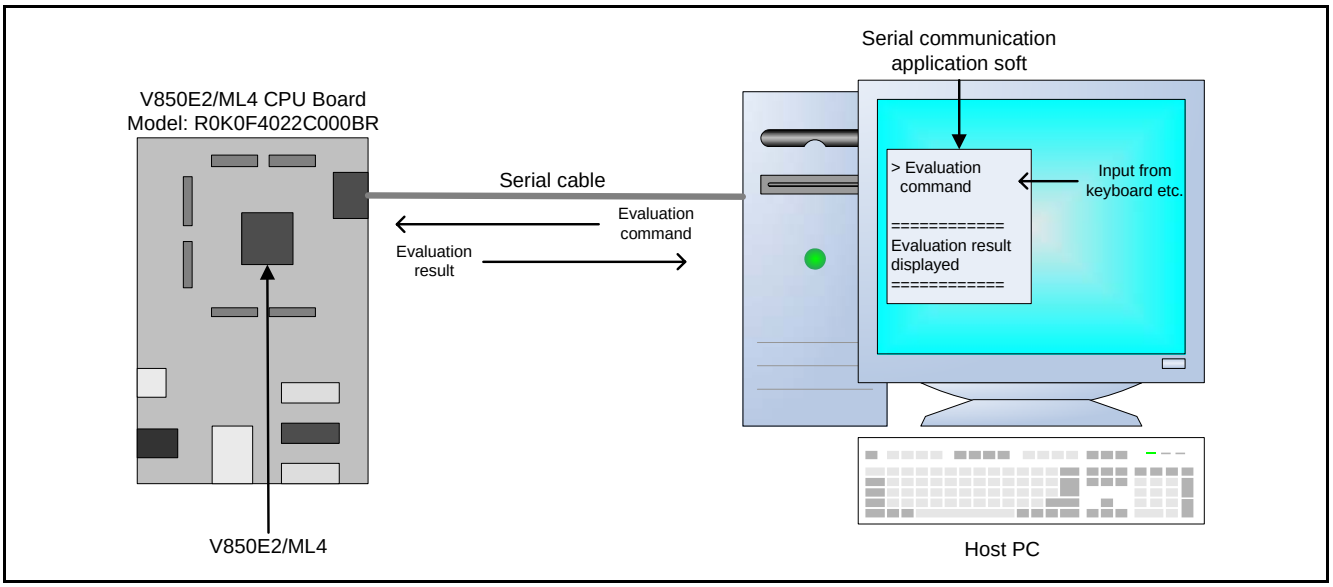

Figure 1.1 System Configuration Diagram

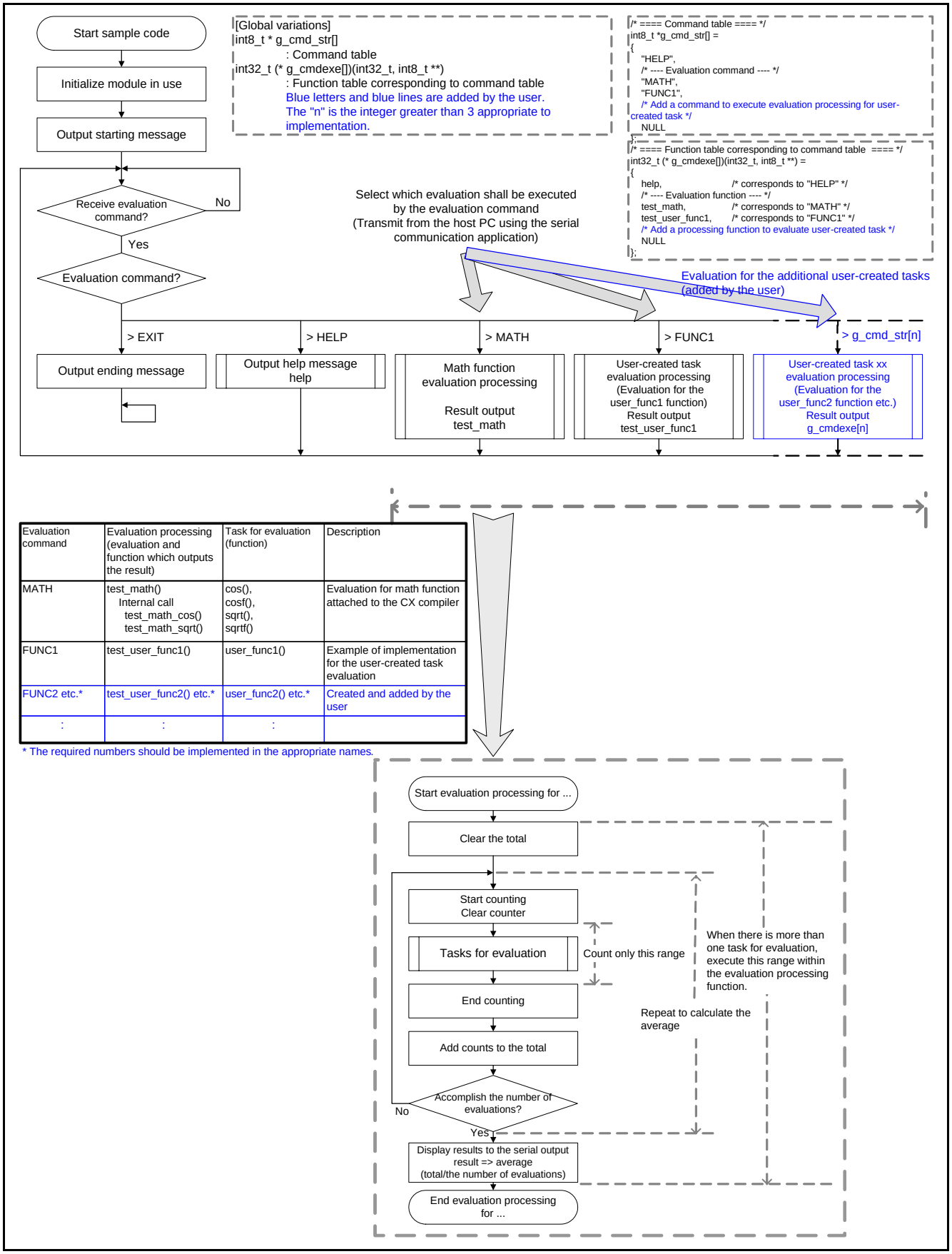

Figure 1.2 Specification Diagram

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	V850E2/ML4
Operating frequency	Internal system clock (f_{CLK}) : 200MHz P bus clock (f_{PCLK}) : 66.667MHz
Operating voltage	Vcc: 3.3V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ Ver.1.02.01
C compiler	Renesas Electronics Corporation CX compiler package Ver.1.21
	Compile option -C f4022 -o DefaultBuild\v850e2ml4_eval.lmf -Xobj_path=Defaultbuild -g -I%ProjectDir%\inc -Xdef_ver +Xide -Xmap=DefaultBuild\v850e2ml4_eval.map -Xhex=DefaultBuild\v850e2ml4_eval.hex
Operating mode	Normal operating mode
Sample code version	1.00
Board used	R0K0F4022C000BR
Tool used	Serial communication application

3. Hardware

3.1 Pins Used

Table 3.1 lists the Pins Used and Their Functions.

Table 3.1 Pins Used and Their Functions

Pin Name	I/O	Function
P2_13/TXD0F	**Output**	Used as output to the serial port
P2_12/RXD0F	**Input**	Used as input from the serial port

4. Software

4.1 Operation Overview

This sample code uses the timer array unite A (TAUA) to evaluate user-created tasks (functions).

The evaluation command is transferred to the performance evaluation software in which had the user-created tasks embedded from the serial communication application (serial terminal) of the host PC to execute evaluation. During execution of the user-created tasks, the number of PCLK cycles required for the TAUA is counted, and the number of CPU clock cycles is calculated to evaluate the performance. The result of the performance evaluation will be displayed in the serial terminal by transmitting from the serial output.

Figure 4.1 shows the Performance Evaluation Diagram.

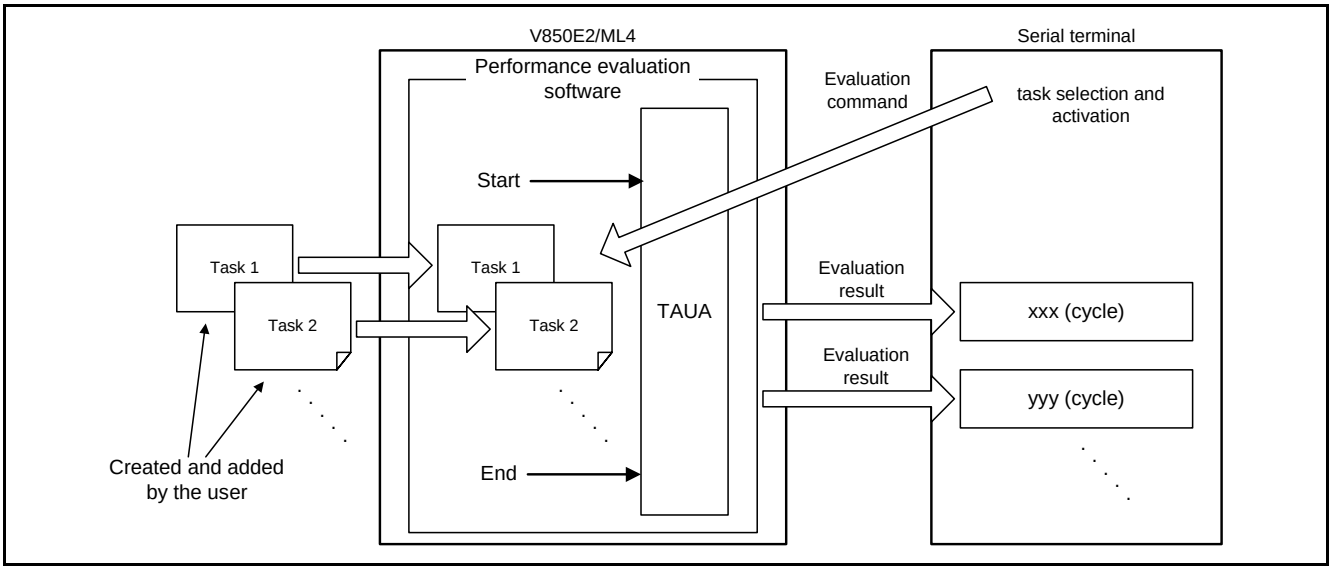

Figure 4.1 Performance Evaluation Diagram

4.2 File Composition

Table 4.1 lists the Files Used in the Sample Code. Files not generated by the integrated development environment should not be listed in this table.

Table 4.1 Files Used in the Sample Code

File Name	Outline	Remarks
main.c	Main processing module	
io_taua0_timer.c	Timer processing module	
io_uartj0_stdio.c	Serial I/O processing module	
test_math.c	Math functions evaluation module	
test_user.c	User-created task evaluation module	
user_func1.c	User-created function module	
io_taua0_timer.h	Timer processing header	
io_uartj0_stdio.h	Serial I/O processing header	
user_func1.h	User-created function module header	
r_typedefs.h	Fixed width integral types definition header	

4.3 Constants

Table 4.2 lists the Constants Used in the Sample Code.

Table 4.2 Constants Used in the Sample Code

Constant Name	Setting Value	Contents
MAX_ARGNUM	8	Maximum number of arguments
MAX_ARGLENGTH	256	Maximum length of argument characters
BUF_SIZE_WAIT	256	Buffer size waiting for string input
BUF_SIZE_UARTJ0_STDOUT	512	Output buffer size
BUF_SIZE_UARTJ0_STDIN	256	Input buffer size
PI	3.141592653589	Pi
EVAL_NUM	256	Number of evaluations executed for MATH command processing
EVAL_TIMES	10	Number of evaluations executed for FUNC1 command processing
LOOP_TIMES	100	Number of empty loops executed during the user-created task processing

4.4 Variables

Table 4.3 lists the Global Variables.

Table 4.3 Global Variables

Type	Variable Name	Contents	Function Used
int8_t	g_buf_wait[WAIT_BUF_SIZE]	Buffer waiting for string input	Main, cmdline_pars
int8_t	g_arg [MAX_ARGNUM][MAX_ARGLENGTH]	Storage area for command string group	command_exe, cmdline_split, cmdline_pars
int8_t	∗ g_cmd_str[]	Command table	cmdline_pars
int32_t	(∗ g_cmdexe[])(int32_t, int8_t ∗∗)	Processing function table corresponding to command table	cmdline_pars
double	g_buf_double_result[EVAL_NUM]	Storage buffer for calculation result of cos function and sqrt function	test_math, test_math_cos, test_math_sqrt
float	g_buf_float_result[EVAL_NUM]	Storage buffer for calculation result of cosf function and sqrtf function	test_math, test_math_cos, test_math_sqrt
uint32_t	g_cnt_int_taua0	Counter for measuring the number of interrupt processing executions	io_int_taua0_count_time, io_taua0_start_timer, io_taua0_get_counter
uint8_t	g_buf_uartj0_stdout [UARTJ0_STDOUT_BUF_SIZE]	Output data buffer	io_uartj0_printf
uint8_t	g_buf_uartj0_stdin [UARTJ0_STDIN_BUF_SIZE]	Input data buffer	io_int_uartj0_recv, io_uartj0_fgets
int32_t	g_cnt_uartj0_stdin	Counter for the number of input data	io_int_uartj0_recv, io_uartj0_fgets

4.5 Functions

Table 4.4 lists the Functions.

Table 4.4 Functions

Function Name	Outline
Main	Main processing
command_exe	Execution processing for command line string
cmdline_split	Split processing for command line string
cmdline_pars	Analytical processing for command string group
Help	Help command processing
io_init_uartj0	UARTJ0 initialization
io_int_uartj0_recv	UARTJ0 receive interrupt processing
io_uartj0_printf	UARTJ0 specified format serial output processing
io_uartj0_fgets	UARTJ0 line reading serial input processing
io_init_taua0	TAUA0 initialization
io_int_taua0_count_timer	TAUA0 interrupt processing
io_taua0_start_timer	TAUA0 timer counting operation start processing
io_taua0_stop_timer	TAUA0 timer counting operation stop processing
io_taua0_get_counter	TAUA0 timer counting acquisition processing
test_math	Evaluation processing for math function library speed
test_math_cos	Evaluation processing for cosine calculation speed
test_math_sqrt	Evaluation processing for square root calculation speed
test_user_func1	Evaluation processing for user-created function processing speed
user_func1	User-created function processing

4.6 Function Specifications

The following tables list the sample code function specifications.

Main	
Outline	Main processing
Header	
Declaration	void main(void)
Description	After initialization, waits and executes the command input by calling the command_exe function.
Arguments	None
Return Value	None

command_exe	
Outline	Execution processing for command line string
Header	
Declaration	int32_t command_exe(int8_t * buf)
Description	Executes a processing which corresponds to the command line string specified by the argument. Execute the split processing of the command line string by calling the cmdline_split function, and stores the string into the command string and the string-array (command string group) which includes more than one argument string. Then analyzes the command string group by calling the cmdline_pars function.
Arguments	int8_t * buf : Command line string
Return Value	>0 : Depends on command processing 0 : Normal end -1 : Detects EXIT command

cmdline_split	
Outline	Split processing for command line string
Header	
Declaration	int32_t cmdline_split(int8_t * cmdline, int8_t * argv[])
Description	Splits the command line string specified by the cmdline with a space, and the split command string and argument string can be stored into the array argv up to the number of MAX_ARGNUM. When the beginning of the command line string specified by the cmdline has '>', split processing will be started from the character secondary to the '>'. When there are more than two consecutive spaces during split processing, the spaces shall be considered as one space. But if the string is surrounded by double quotes, it is considered as a single string. Also, returns the number of stored strings.
Arguments	int8_t * cmdline : Command line string int8_t * argv[] : Address to store the split result (command string group)
Return Value	Number of strings in the command string group

cmdline_pars	
Outline	Analytical processing for command string group
Header	
Declaration	int32_t cmdline_pars(int32_t argc, int8_t * argv[])
Description	Analyzes the command string specified by the argument argv and executes corresponding command processing function. The command to be executed is registered in the command table g_cmd_str[]. When adding or deleting a command, change the command table g_cmd_str[] and the function table g_cmdexe[] corresponding to the command table.
Arguments	int32_t argc : Number of strings in the command string group int8_t * argv[] : Command string group
Return Value	>0 : Depends on command processing 0 : Normal end other than command table -1 : Detects EXIT command

Help	
Outline	Help command processing
Header	
Declaration	int32_t help(int32_t argc, int8_t ** argv)
Description	Describes the evaluation command using the UARTJ0 serial output.
Arguments	int32_t argc : Number of strings in the command string group int8_t ** argv : Command string group
Return Value	0 : Normal end -1 : Error

io_init_uartj0	
Outline	UARTJ0 initialization
Header	
Declaration	void io_init_uartj0(void)
Description	Initializes the UARTJ0 to set it for the serial I/O.
Arguments	None
Return Value	None

io_int_uartj0_recv	
Outline	UARTJ0 receive interrupt processing
Header	
Declaration	void io_int_uartj0_recv(void)
Description	Stores the UARTJ0 received data to the receive data buffer.
Arguments	None
Return Value	None

io_uartj0_printf	
Outline	UARTJ0 specified format serial output processing
Header	
Declaration	int32_t io_uartj0_printf(const int8_t format[], ...)
Description	Executes serial output using the UARTJ0 according to the format specified by the argument.
Arguments	const int8_t format[], ... : Output format string and data based on the format
Return Value	Number of output bytes

io_uartj0_fgets	
Outline	UARTJ0 line reading serial input processing
Header	
Declaration	int8_t * io_uartj0_fgets(int8_t * s, int32_t n, FILE * stream)
Description	Reads a string one line by the UARTJ0 serial input.
Arguments	int8_t * s : Address to store the read data int32_t n : Designation of the read size FILE * stream : File pointer (stdin fixed)
Return Value	NULL : End without doing anything when the value of argument stream is not stdin. Other than NULL : The value of arguments (Normal end)

io_init_taua0	
Outline	TAUA0 initialization
Header	
Declaration	void io_init_taua0(void)
Description	Initializes the TAUA0 to use it as the measurement timer for the PCLK clock cycle counts.
Arguments	None
Return Value	None

io_int_taua0_count_timer	
Outline	TAUA0 interrupt processing
Header	
Declaration	void io_int_taua0_count_timer(void)
Description	Increments the measurement counter for the number of interrupt processing executions.
Arguments	None
Return Value	None

io_taua0_start_timer	
Outline	TAUA0 timer counting operation start processing
Header	
Declaration	void io_taua0_start_timer(void)
Description	Clears the measurement counter for the number of interrupt processing executions, and starts the TAUA0 timer counting operation.
Arguments	None
Return Value	None

io_taua0_stop_timer	
Outline	TAUA0 timer counting operation stop processing
Header	
Declaration	void io_taua0_stop_timer(void)
Description	Stops the TAUA0 timer counting operation.
Arguments	None
Return Value	None

io_taua0_get_counter	
Outline	TAUA0 timer counting acquisition processing
Header	
Declaration	uint32_t io_taua0_get_counter(void)
Description	Calculates the number of CPU clock cycles counted by the measurement counter for the number of interrupt processing execution and the TAUAT0 register during the timer counting operation.
Arguments	None
Return Value	Number of cycles counted during the timer counting

test_math		
Outline	Evaluation processing for math function library speed	
Header		
Declaration	int32_t test_math (int32_t argc, int8_t ** argv)	
Description	Evaluates the math function library speed. Calls the evaluation processing function for cosine calculating speed and the evaluation processing function for square root calculation speed.	
Arguments	int32_t argc	: Number of strings in the command string group
	int8_t ** argv	: Command string group
Return Value	1	

test_math_cos	
Outline	Evaluation processing for cosine calculation speed
Header	
Declaration	void test_math_cos (void)
Description	Evaluates the cosine calculating speed. Executes the cos function and cosf function of the MATH library for 256 times, and calculates the average of processing time every time being measured by the TAUA (CPU clock cycles). Outputs the calculation results by calling the io_uartj0_printf function.
Arguments	None
Return Value	None

test_math_sqrt	
Outline	Evaluation processing for square root calculation speed
Header	
Declaration	void test_math_sqrt (void)
Description	Evaluates the square root calculation speed. Executes the sqrt function and sqrtf function of the MATH library for 256 times, and calculates the average of processing time every time being measured by the TAUA (CPU clock cycles). Outputs the calculation results by calling the io_uartj0_printf function.
Arguments	None
Return Value	None

test_user_func1	
Outline	Evaluation processing for user-created function processing speed
Header	
Declaration	int32_t test_user_func1 (int32_t argc, int8_t * argv[])
Description	Executes the user_func1 function for ten times, and calculates the average of processing time every time being measured by the TAUA (CPU clock cycles). Outputs the calculation results by calling the io_uartj0_printf function.
Arguments	int32_t argc : Number of strings in the command string group int8_t ** argv : Command string group
Return Value	1

user_func1	
Outline	User-created function processing
Header	
Declaration	void user_func1(void)
Description	Example of implementation for the user-created function. In the sample code, the empty loop will be executed for 100 times.
Arguments	None
Return Value	None

4.7 Flowcharts

4.7.1 Main Processing

Figure 4.2 shows the Main Processing.

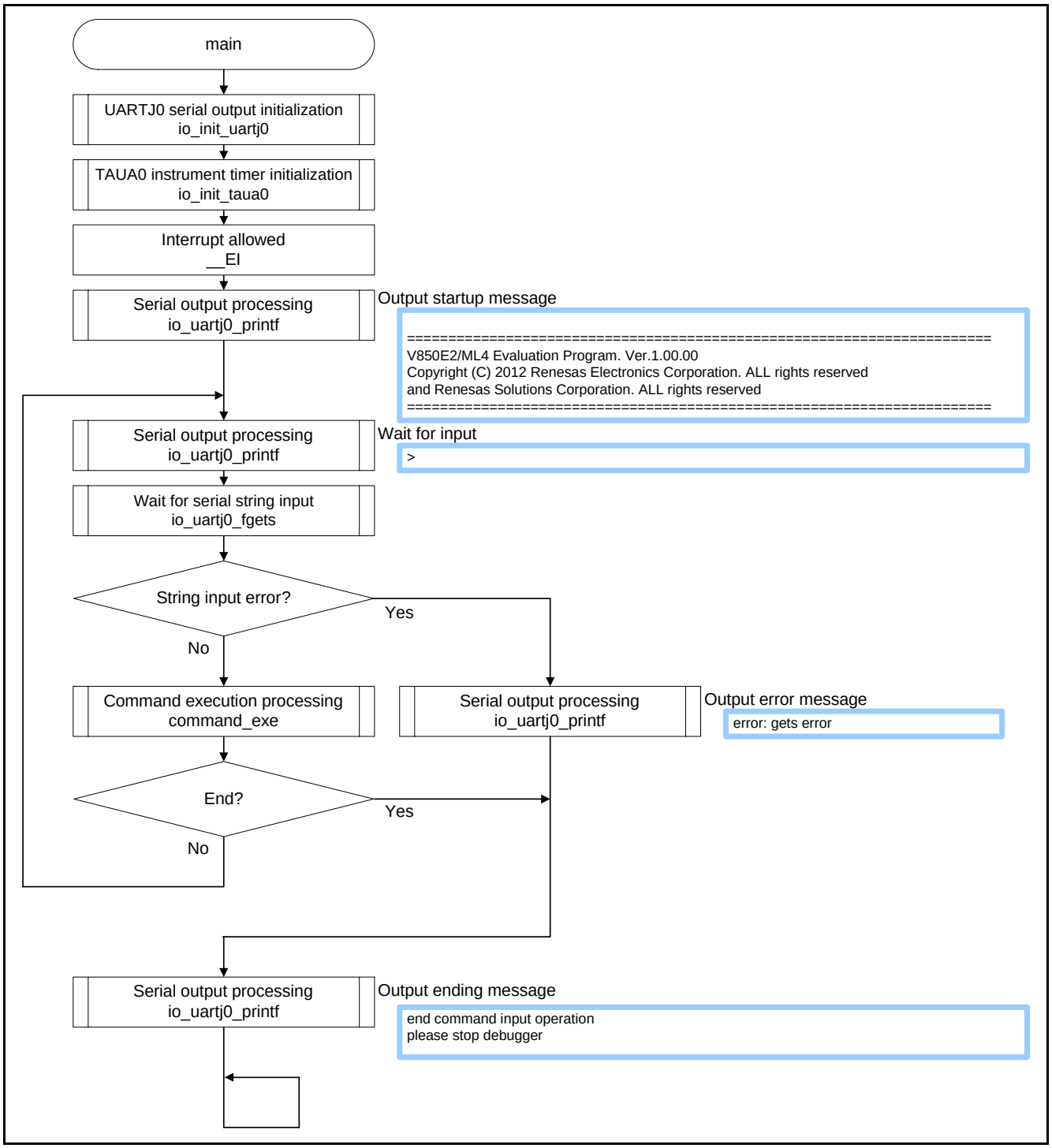

Figure 4.2 Main Processing

4.7.2 Execution Processing for Command Line String

Figure 4.3 shows the Execution Processing for Command Line String.

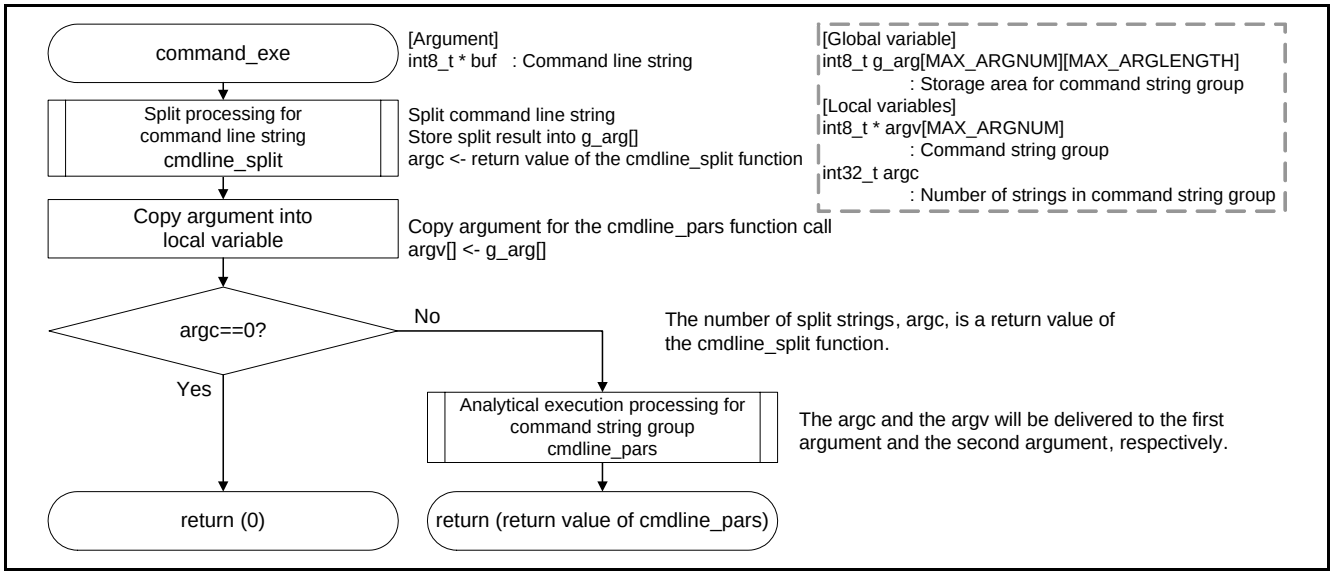

Figure 4.3 Execution Processing for Command Line String

4.7.3 Split Processing for Command Line String

Figure 4.4 and Figure 4.5 show the Split Processing for Command Line String

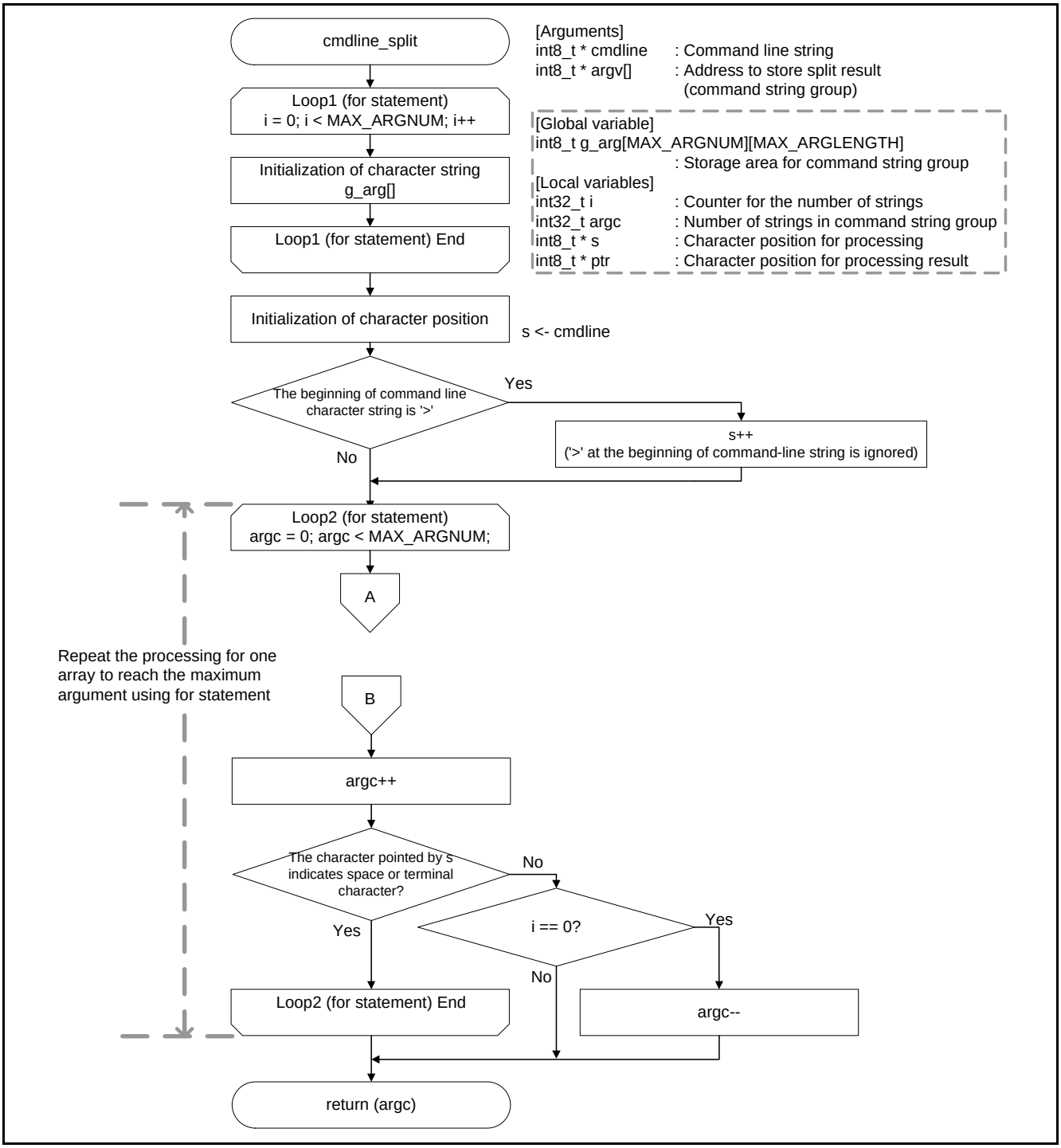

Figure 4.4 Split Processing for Command Line String

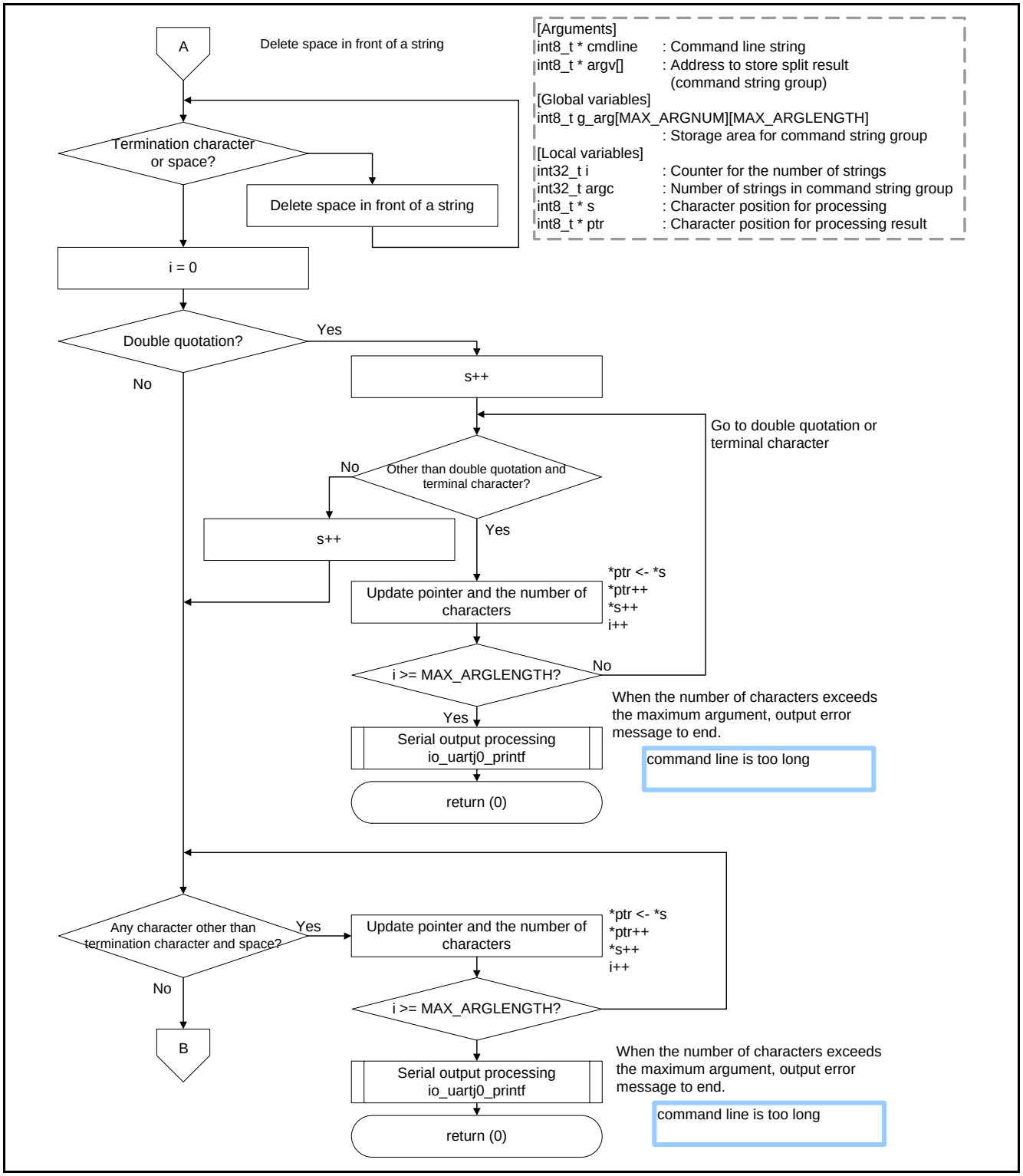

Figure 4.5 Split Processing for Command Line String

4.7.4 Analytical Processing for Command String Group

Figure 4.6 shows the Analytical Processing for Command String Group.

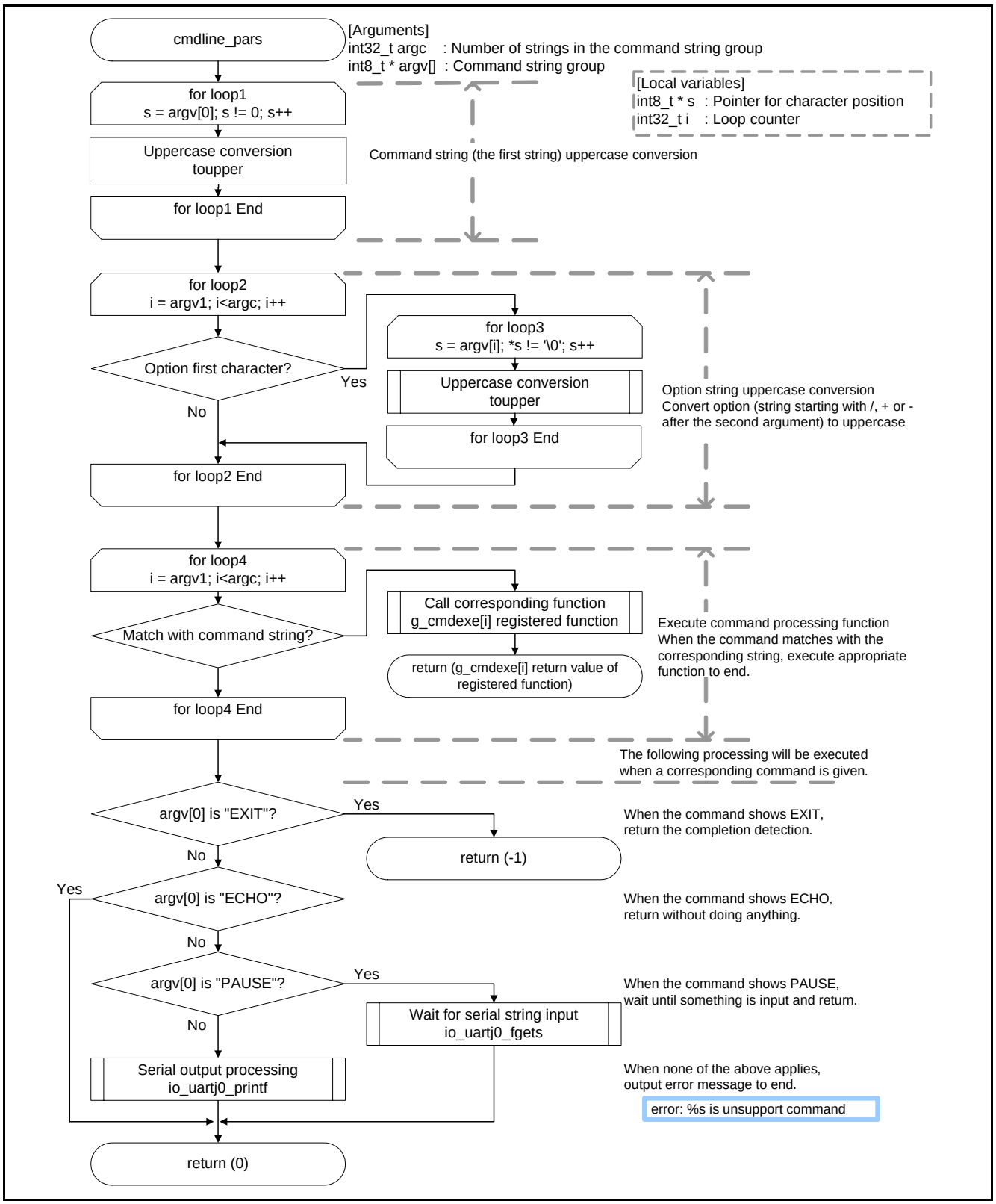

Figure 4.6 Analytical Processing for Command String Group

4.7.5 Help Command Processing

Figure 4.7 shows the Help Command Processing.

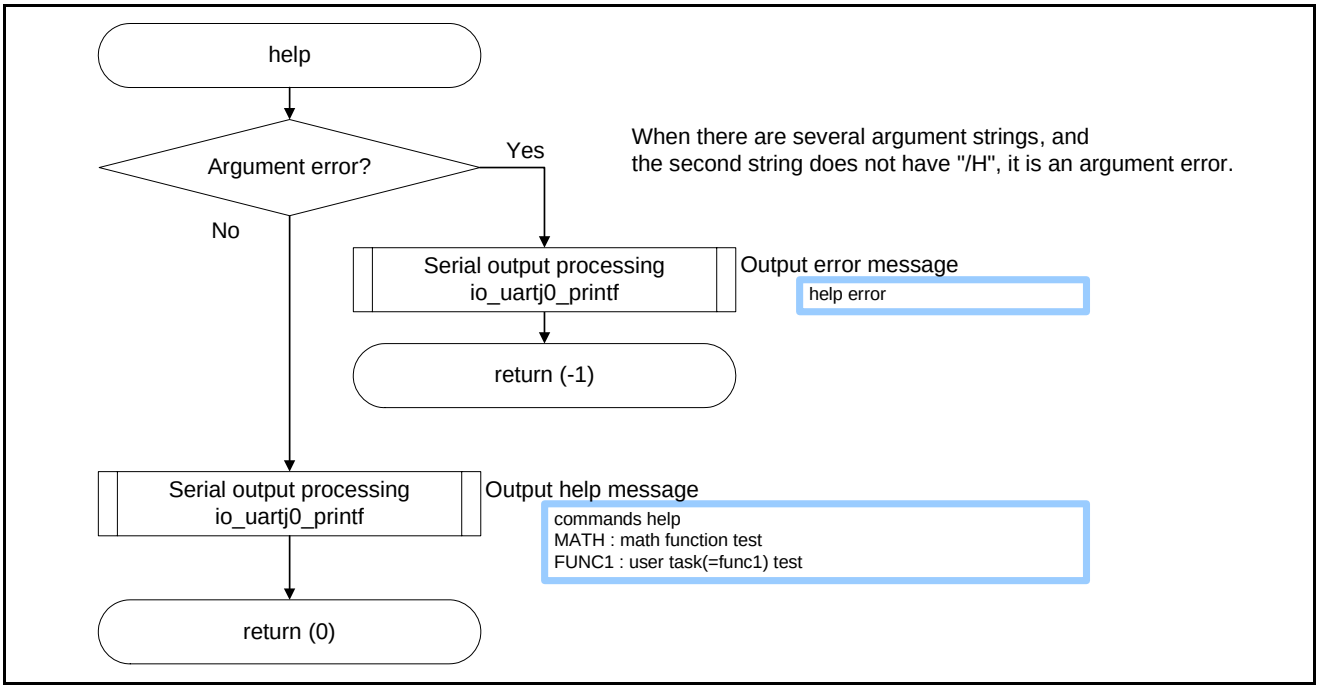

Figure 4.7 Help Command Processing

4.7.6 UARTJ0 Initialization

Figure 4.8 shows the UARTJ0 Initialization.

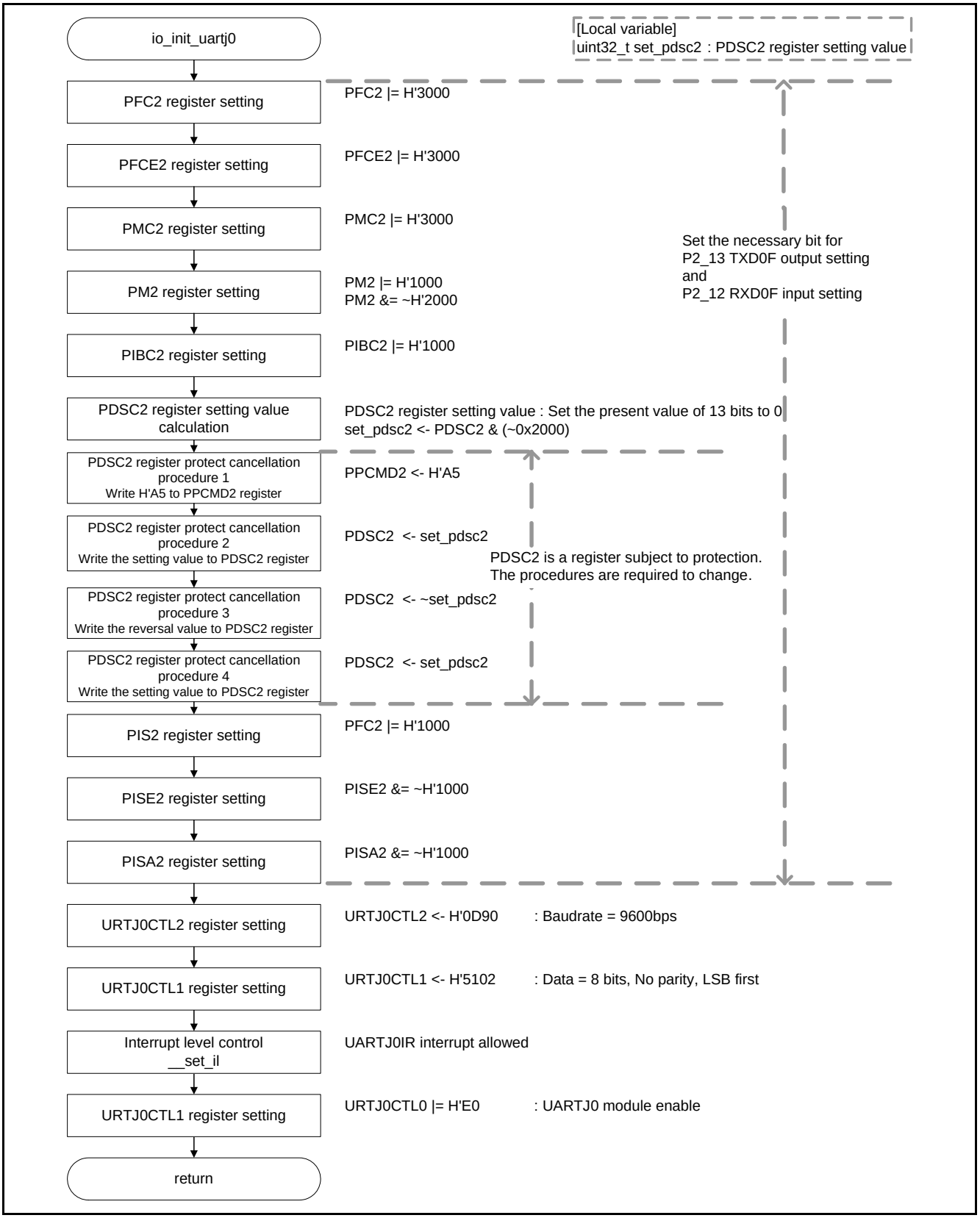

Figure 4.8 UARTJ0 Initialization

4.7.7 UARTJ0 Receive Interrupt Processing

Figure 4.9 shows the UARTJ0 Receive Interrupt Processing.

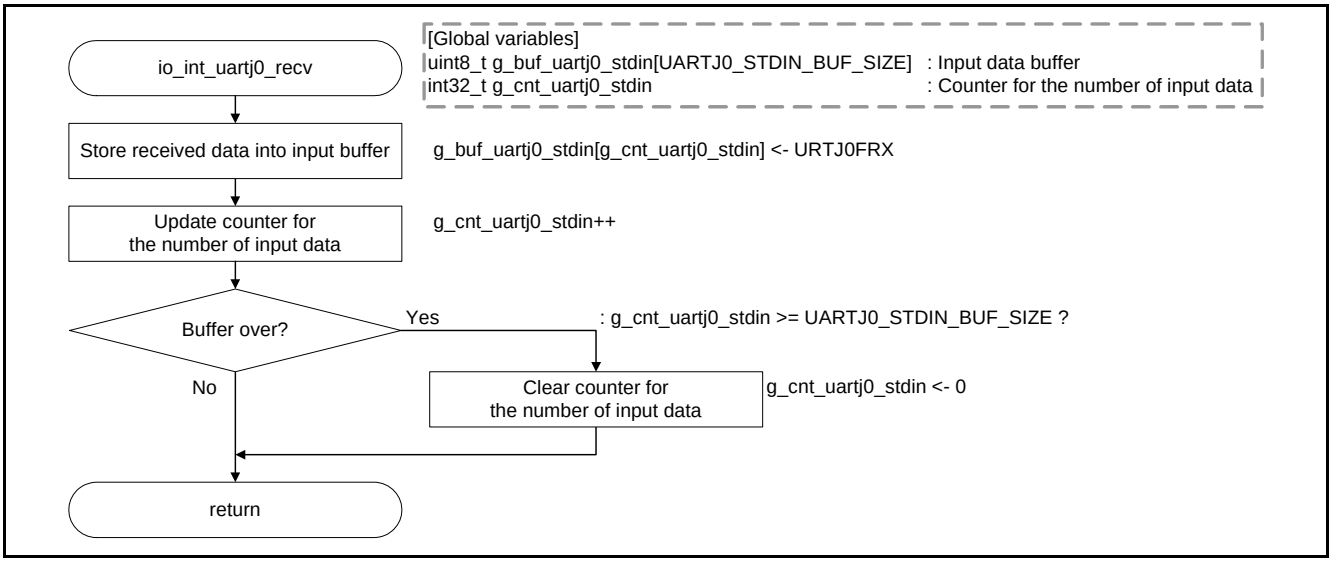

Figure 4.9 UARTJ0 Receive Interrupt Processing

4.7.8 UARTJ0 Specified Format Serial Output Processing

Figure 4.10 shows the UARTJ0 Specified Format Serial Output Processing.

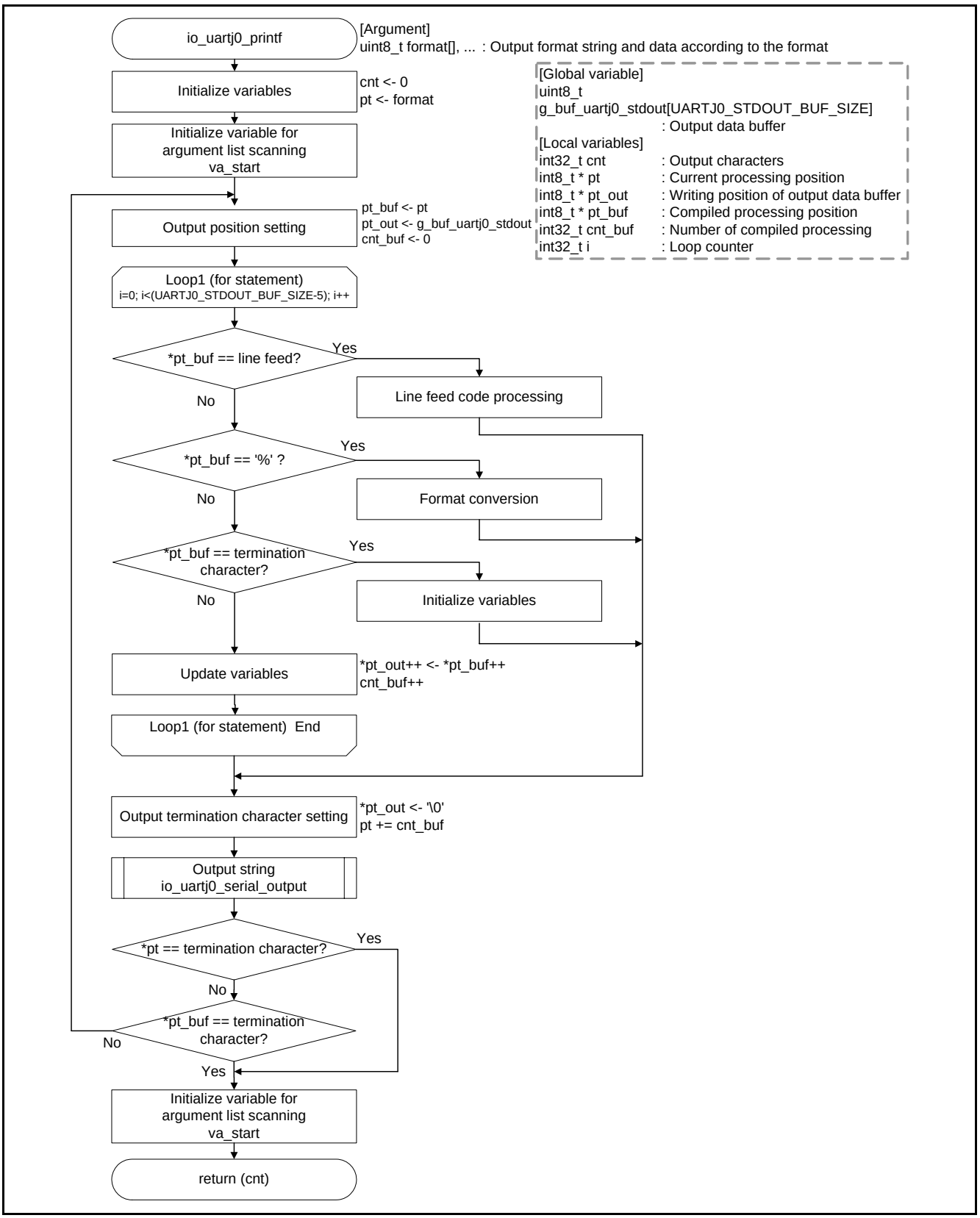

Figure 4.10 UARTJ0 Specified Format Serial Output Processing

4.7.9 UARTJ0 Line Reading Serial Input Processing

Figure 4.11 shows the UARTJ0 Line Reading Serial Input Processing.

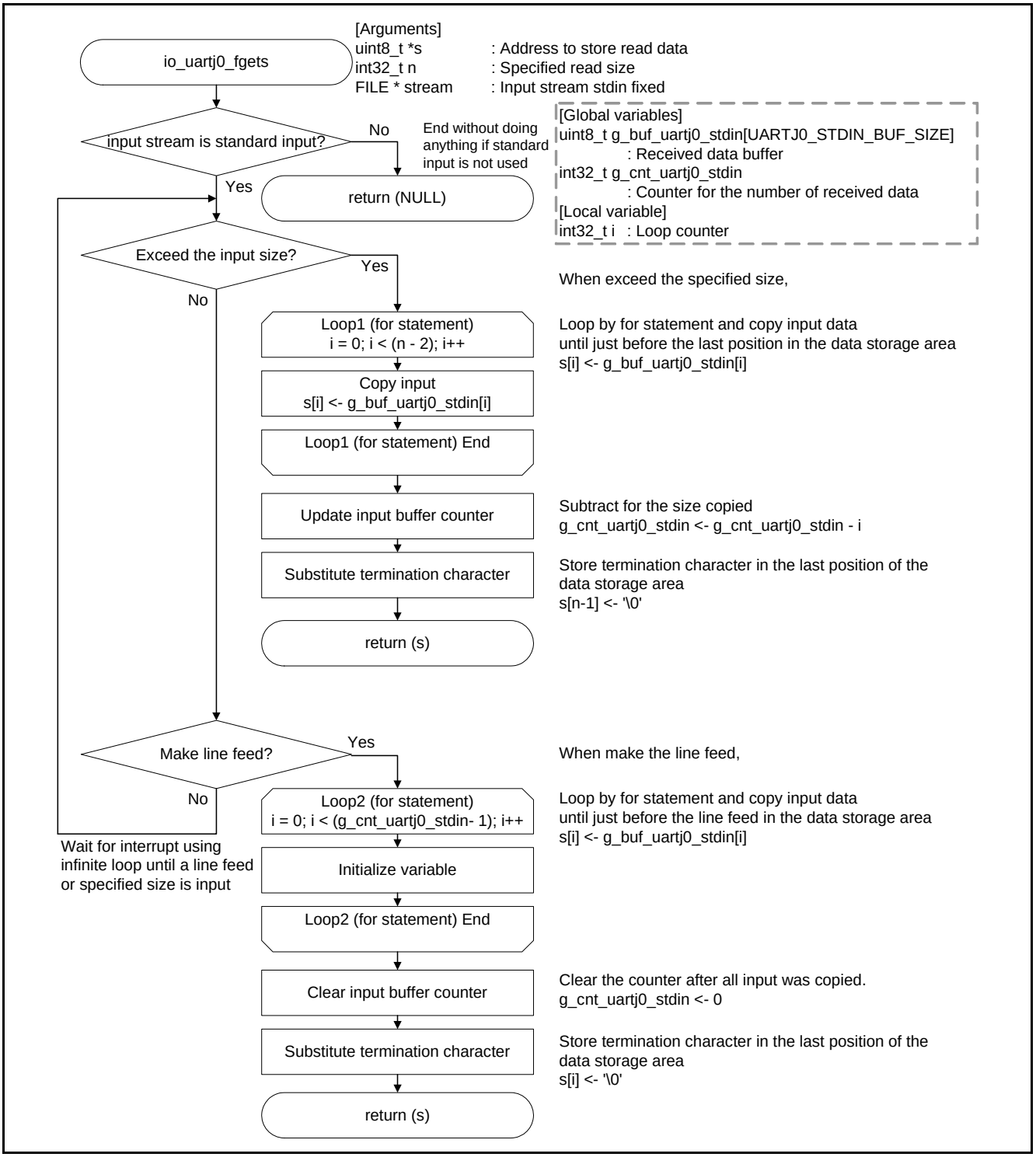

Figure 4.11 UARTJ0 Line Reading Serial Input Processing

4.7.10 TAUA0 Initialization

Figure 4.12 shows the TAUA0 Initialization.

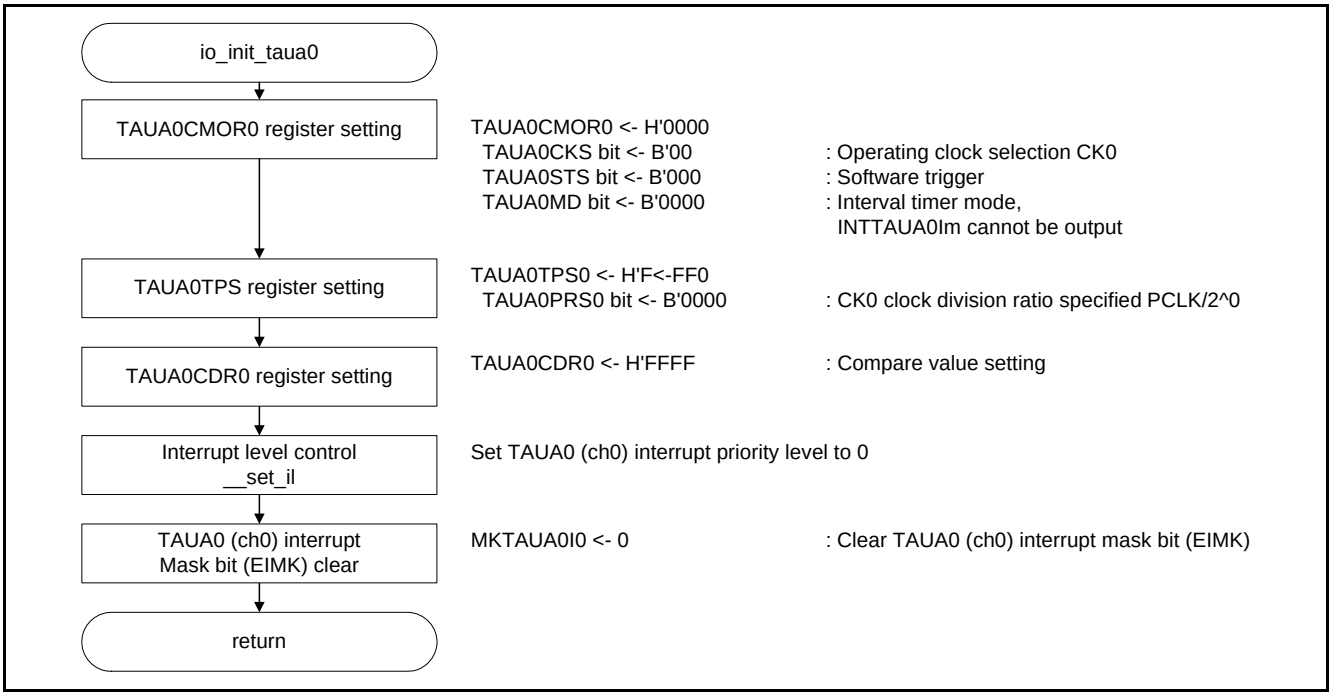

Figure 4.12 TAUA0 Initialization

4.7.11 TAUA0 Interrupt Processing

Figure 4.13 shows the TAUA0 Interrupt Processing.

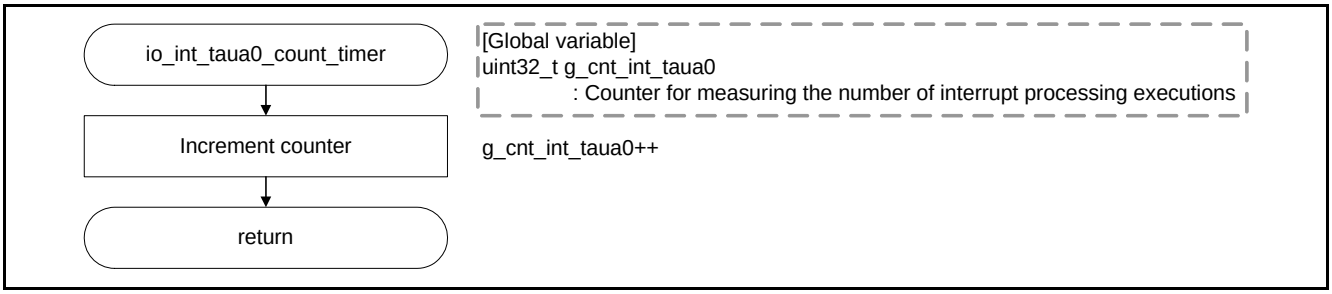

Figure 4.13 TAUA0 Interrupt Processing

4.7.12 TAUA0 Timer Counting Operation Start Processing

Figure 4.14 shows the TAUA0 Timer Counting Operation Start Processing.

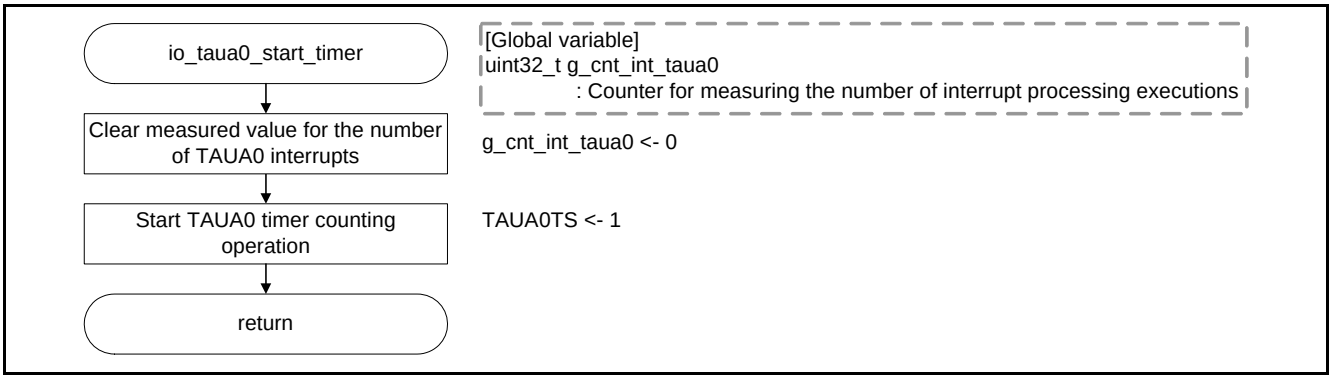

Figure 4.14 TAUA0 Timer Counting Operation Start Processing

4.7.13 TAUA0 Timer Counting Operation Stop Processing

Figure 4.15 shows the TAUA0 Timer Counting Operation Stop Processing.

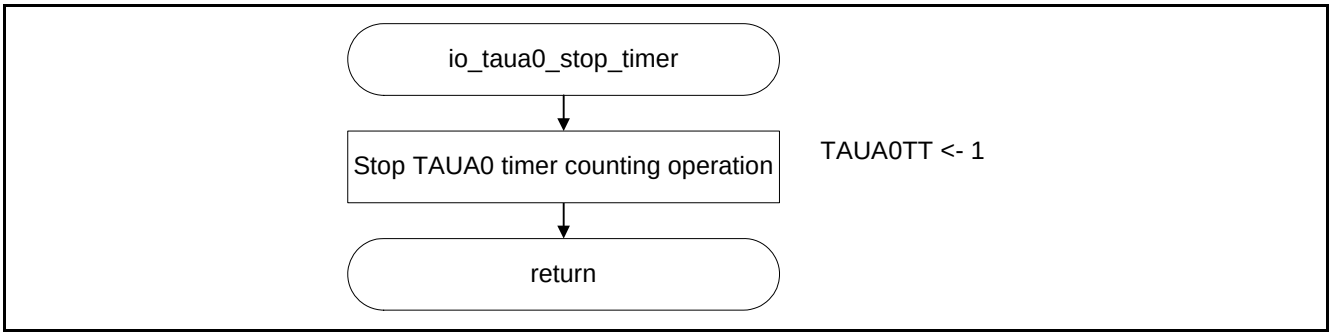

Figure 4.15 TAUA0 Timer Counting Operation Stop Processing

4.7.14 TAUA0 Timer Counting Acquisition Processing

Figure 4.16 shows the TAUA0 Timer Counting Acquisition Processing.

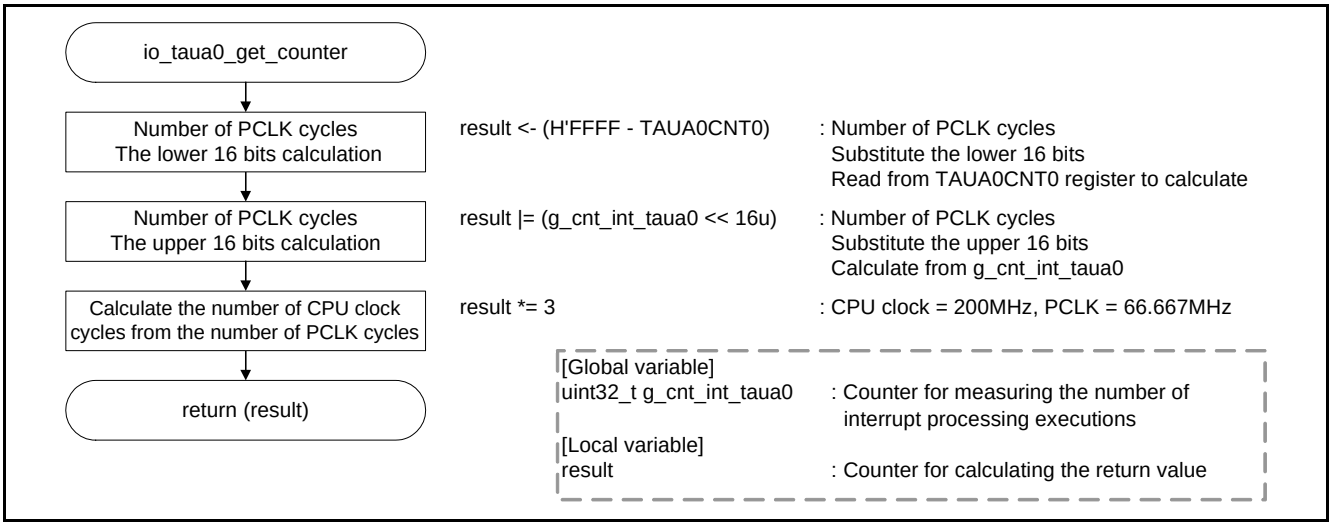

Figure 4.16 TAUA0 Timer Counting Acquisition Processing

4.7.15 Evaluation Processing for Math Function Library Speed

Figure 4.17 shows the Evaluation Processing for Math Function Library Speed.

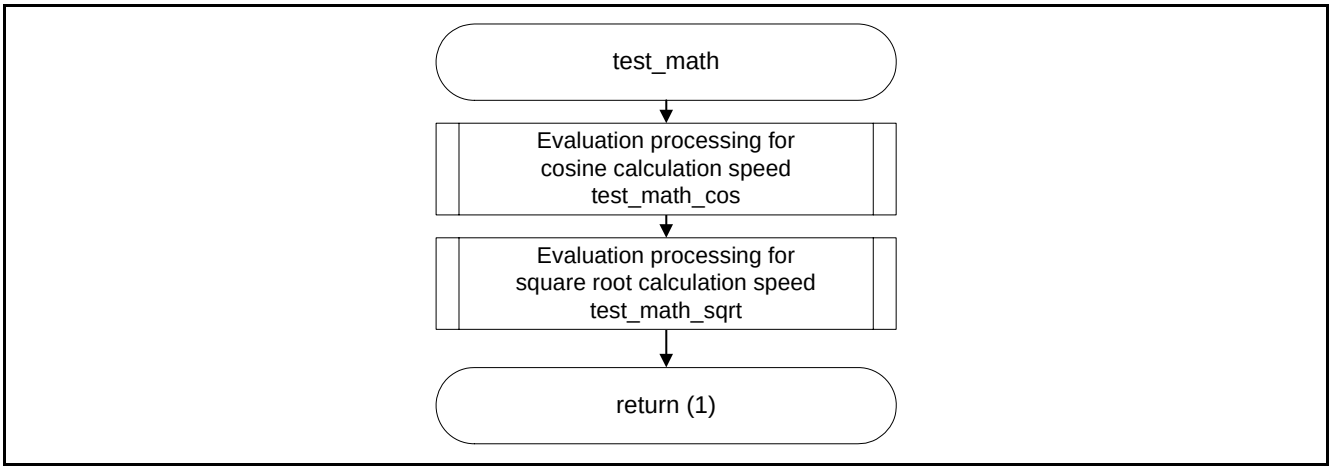

Figure 4.17 Evaluation Processing for Math Function Library Speed

4.7.16 Evaluation Processing for Cosine Calculation Speed

Figure 4.18 shows the Evaluation Processing for Cosine Calculation Speed.

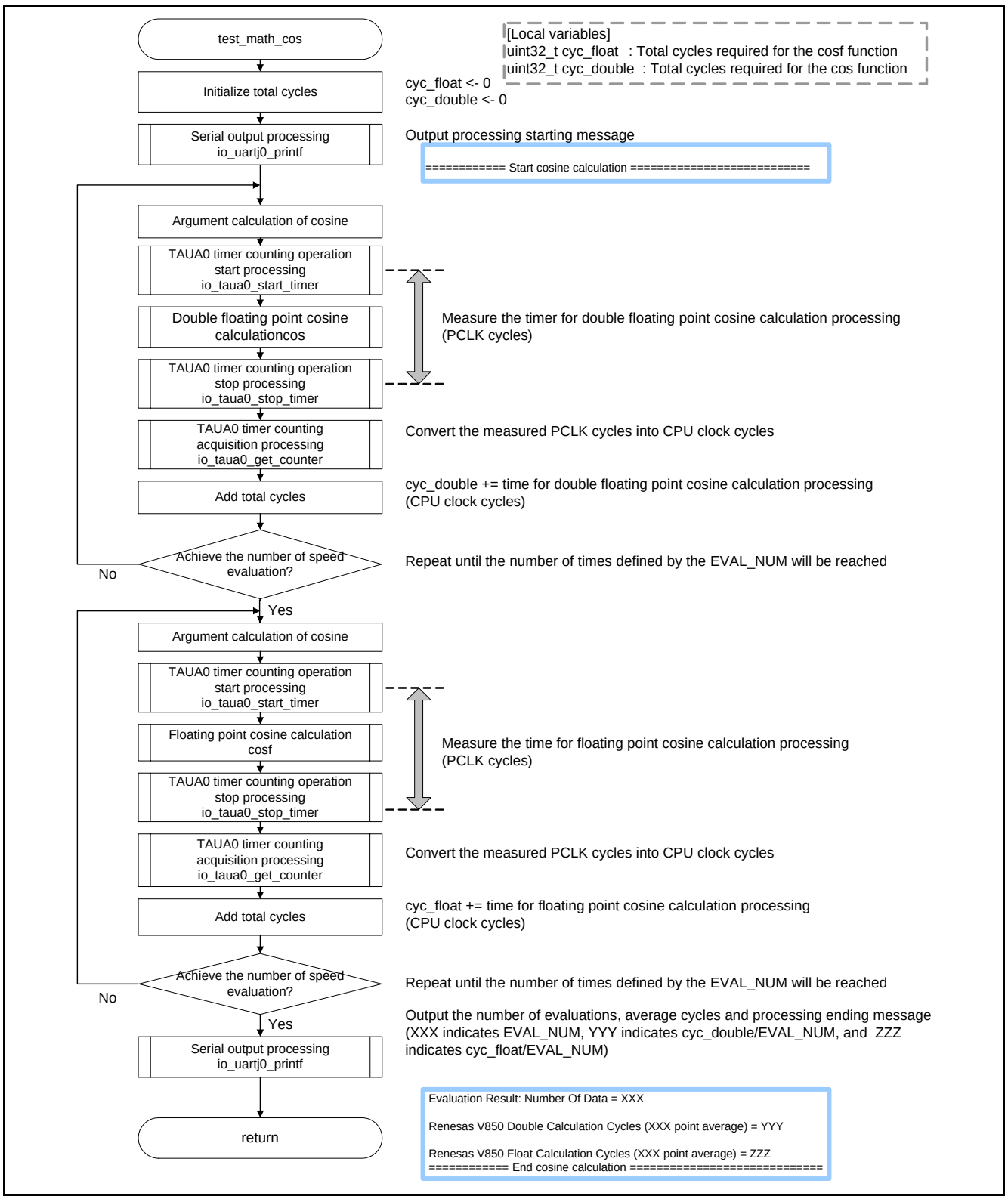

Figure 4.18 Evaluation Processing for Cosine Calculation Speed

4.7.17 Evaluation Processing for Square Root Calculation Speed

Figure 4.19 shows the Evaluation Processing for Square Root Calculation Speed.

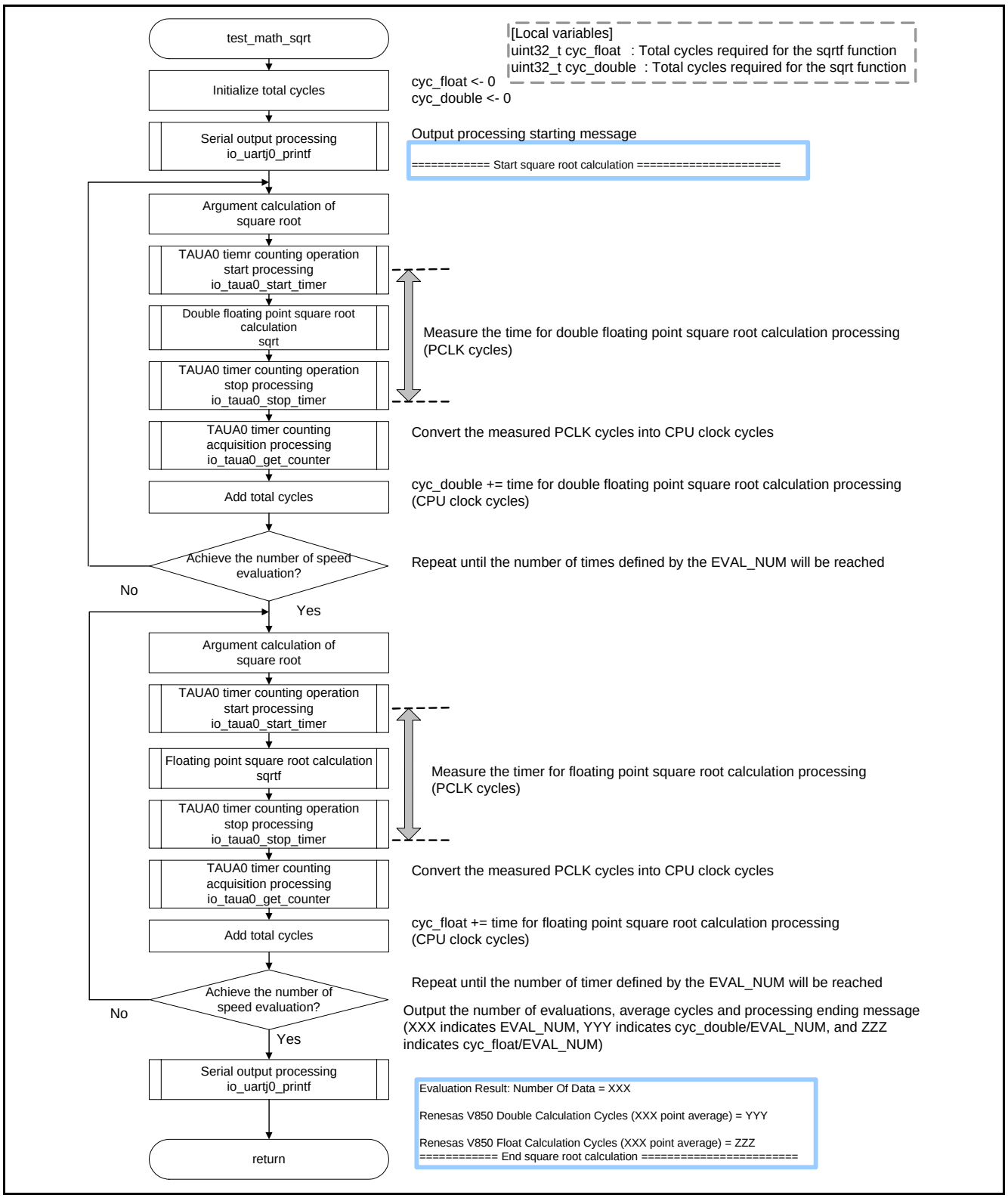

Figure 4.19 Evaluation Processing for Square Root Calculation Speed

4.7.18 Evaluation Processing for User-Created Function Processing Speed

Figure 4.20 shows the Evaluation Processing for User-Created Function Processing Speed.

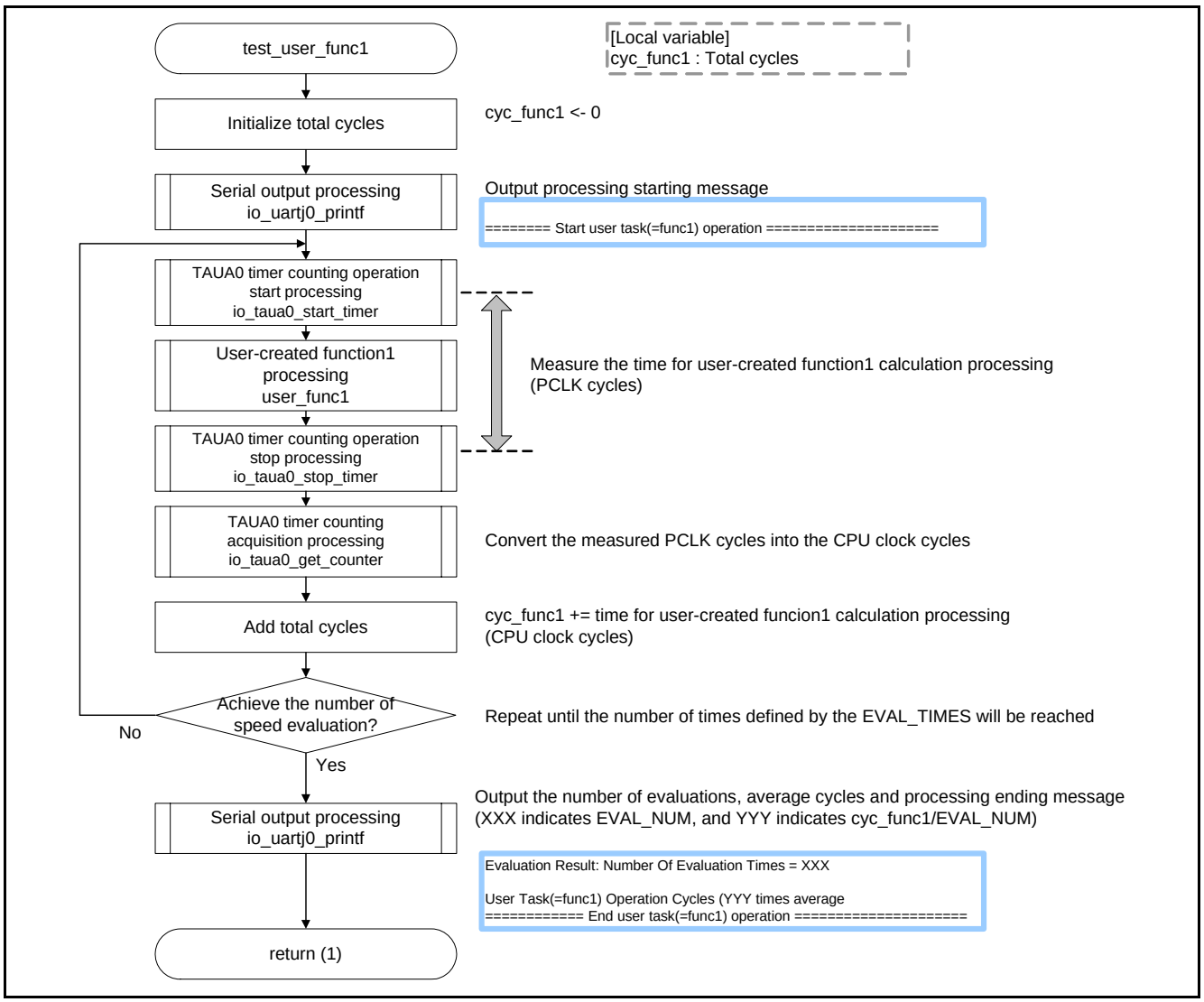

Figure 4.20 Evaluation Processing for User-Created Function Processing Speed

4.7.19 User-Created Function Processing

Figure 4.21 shows the User-Created Function Processing.

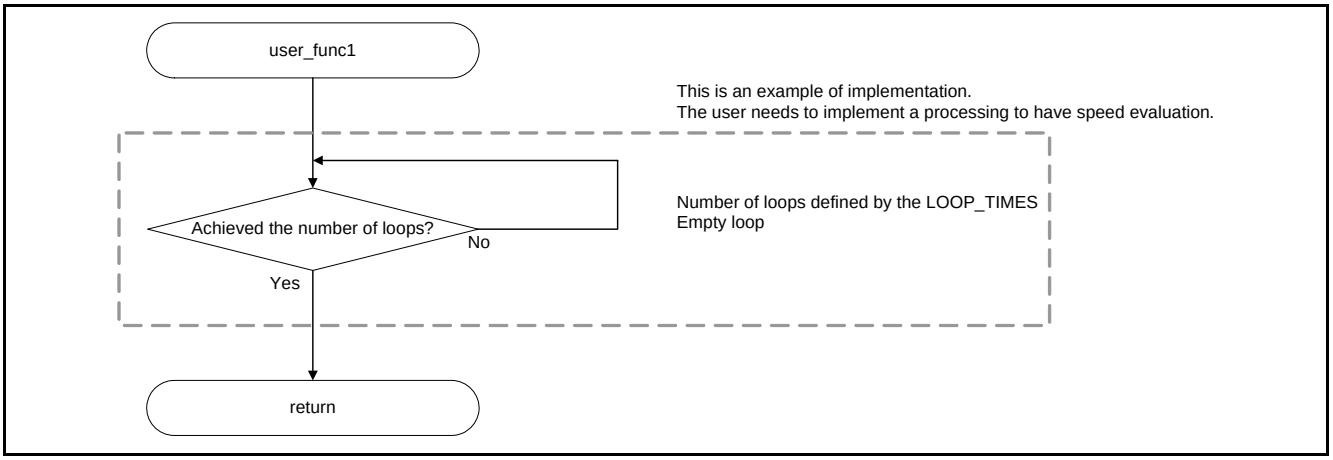

Figure 4.21 User-Created Function Processing

5. Application Example

This chapter describes the performance evaluation method using sample code and how to add a user-creased task.

5.1 Performance Evaluation

This section describes the procedure for the evaluation.

1. Connect the host CP and the V850E2/ML4 CPU board with the serial cable. If the PC does not have a serial port, a USB serial convertible cable can be used.
2. Activate the serial communication application in the host PC and make communication setting as follows.

   Transfer rate 9600bps, Data length 8, No parity, 1 stop bit, Flow control Xon/Xoff
3. Load and execute the program.
4. After executing the program, the following log will be displayed on the serial communication application.

```
======================================================================
V850E2/ML4 Evaluation Program. Ver.1.00.00
Copyright (C) 2012 Renesas Electronics Corporation. ALL rights reserved
and Renesas Solutions Corporation. ALL rights reserved
======================================================================
>
```

Figure 5.1 Performance Evaluation Activation Log

5. When input "HELP" in the console of the serial communication application, the evaluation command type will be displayed. The following is a display example when 100 times loop processing (=user_func1) are embedded for the math function calculation included in the sample code and the user-created task. Any task the user desires to evaluate and its corresponding command can be added as may be necessary.

```
> HELP
commands help
MATH : math function test
FUNC1 : user task(=func1) test
>
```

Figure 5.2 Performance Evaluation HELP Log

6. When input "MATH" in the console of the serial communication application, it executes the math function calculation using the V850 embedded library (=math.h) included in the sample code, and the number of cycles required for the calculation will be displayed. The following shows the average value when repeating the math function calculation for 256 times using the V850 embedded library.

```
> MATH
============ Start cosine calculation ============================
Evaluation Result: Number Of Data = 256

Renesas V850 Double Calculation Cycles (256 point average) = 341

Renesas V850 Float Calculation Cycles (256 point average) = 75
============ End cosine calculation ==============================

============ Start square root calculation =======================
Evaluation Result: Number Of Data = 256

Renesas V850 Double Calculation Cycles (256 point average) = 56

Renesas V850 Float Calculation Cycles (256 point average) = 38
============ End square root calculation ========================
>
```

Figure 5.3 Performance Evaluation MATH Log

7. When input "FUNC1" in the console of the serial communication application, it executes 100 times loop processing included in the sample code as a user-created task example, and the number of cycles required for the processing will be displayed. The following shows the average value when repeating 100 times loop processing for ten times.

```
> FUNC1
======== Start user task(=func1) operation =====================
Evaluation Result: Number Of Evaluation Times = 10

User Task(=func1) Operation Cycles (10 times average) = 1221
============ End user task(=func1) operation =====================
>
```

Figure 5.4 Performance Evaluation FUNC1 Log

[Note] Precaution for Evaluation

The measurement in the sample code is executed by the timer, but the number of cycles is required for the CPU timer start processing and end processing. Therefore, it is necessary to measure the number of cycles when the task is not executed and deduct it from the number of cycles when the task is executed.

For example, if the EMPTY_LOOP of the test_math.c is validated (comment out #undef EMPTY_LOOP) in the sample code math function calculation, the number of cycles when the task is not executed can be measured.

```
/* ---- Empty loop calculation ---- */
/* Calculation result will be derived by subtraction from empty loop calculation result */
#define EMPTY_LOOP
//#undef EMPTY_LOOP /* Empty loop calculation will be executed when comment out this line */
```

Figure 5.5 Example of Implementation for Empty loop

5.2 Adding A User-Created Task

This section describes how to add a user-created task.

1. Create a task to be added as a function with C language (hereinafter called user_func2).
2. Embed the user_func2 into the test_user.c , along with the user_func1 in the test_user_func1, by executing the io_taua0_start_timer function before the user_func2, and also executing the io_taua0_stop_timer function after the user_func2, which

```
        for (i = 0; i < EVAL_TIMES; i++)
        {
                io_taua0_start_timer();
#ifndef EMPTY_LOOP
                user_func2();
#endif
                io_taua0_stop_timer();

                cyc_func1 += io_taua0_get_counter();
        }
```

Figure 5.6 Example of Adding A User-Created Task

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

User's Manual: Hardware

V850E2/ML4 User's Manual: Hardware (R01UH0262EJ)

The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

CubeSuite+ V1.00.00 Integrated development environment User's Manual: Coding (CX compiler) (R20UT0554EJ)

CubeSuite+ V1.00.00 Integrated development environment User's Manual: Build (CX compiler) (R20UT0558EJ)

The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website

http://www.renesas.com

Inquiries

http://www.renesas.com/contact/

REVISION HISTORY	V850E2/ML4 Application Note Performance Evaluation Software

Rev.	Date	Description	
		Page	Summary
1.00	Aug. 24, 2012	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "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 etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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.

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.0