RL78/G10

Multiplication and Division Program CC-RL

Introduction

This application note explains the program which realizes multiplication and division of unsigned integer using RL78/G10.

A program is shown about three kinds of arithmetic processing (16bit×16bit, 16bit×16bit+32bit, and 32bit÷32bit).

Target Device

RL78/G10

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Contents

1. Specifications

In this application note, an actual program and its usage are shown from the view of operation for three kinds of arithmetic processing subroutines (16bit×16bit, 16bit×16bit+32bit, and 32bit÷32bit).

Table 1.1 shows the arithmetic processing objected in this application note.

Table 1.1 Targeted Arithmetic Processing

Arithmetic Processing	Description
16bit×16bit	Multiplication of unsigned integers of 16-bit data.
16bit×16bit+32bit	Multiply and accumulation of unsigned integers of 16-bit data. The overflow flow is not detected from 32-bit data.
32bit÷32bit	The division processing of unsigned integers of 32-bit data.

2. Operation Check Conditions

The sample code described in this application note has been checked under the conditions listed in the table below.

Table 2.1 Operation Check Conditions

Item	Description
Microcontroller used	RL78/G10 (R5F10Y16)
Operating frequency	· High-speed on-chip oscillator (HOCO) clock: 20MHz · CPU/peripheral hardware clock: 20MHz
Operating voltage	5.0 V (can run on a voltage range of 2.9 V to 5.5 V.) SPOR detection voltage: Rising edge voltage: 2.90V : Falling edge voltage: 2.84V
Integrated development environment (CS+)	CS+ for CC V3.01.00 from Renesas Electronics Corp.
Assembler (CS+)	CC-RL V1.01.00 from Renesas Electronics Corp.
Integrated development environment (e^2 studio)	e^2 studio V4.0.2.008 from Renesas Electronics Corp.
Assembler (e^2 studio)	CC-RL V1.01.00 from Renesas Electronics Corp.
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.21.3 from IAR Systems.
Assembler (IAR)	IAR Assembler for Renesas RL78 V4.21.2.2420 from IAR Systems.
Operating environment	RL78/G10 Simulator

3. Description of Software

3.1 Operation Outline

In the arithmetic processing of this application note, specified operation is performed to the data stored in the variable arranged to a short direct area, and the result is set to the variable arranged to the short direct area.

3.2 The Way of Thinking of Multiplication

Calculation is performed by dividing large data into a small digit as well as the way of the usual computation on paper.

In this case, attention is necessary for digit place (figure). Here, a calculation of hexadecimal numbers of A1A2A3A4 x B1B2B3B4 is shown as an example. As the calculation method, the 8-bit multiply instruction "a MULU X command" is used. Calculates in 4 steps of every 2 digits (8 bits) (multiplication); A1A2 × B1B2, A1A2 × B3B4, A3A4 × B1B2, A3A4 × B3B4, and 4 times of calculation results are added with attention to digit place.

This system is shown in Figure 3.1. Thus, an answer can be obtained by adding in consideration of the digit of 4 times of calculation results. Here, although a multiplicand and a multiplier are 4 digits (16 bits), if they are beyond 4 digits (16 bits), they are calculable similarly by adding the calculation results of every double figure with attention to digit place.

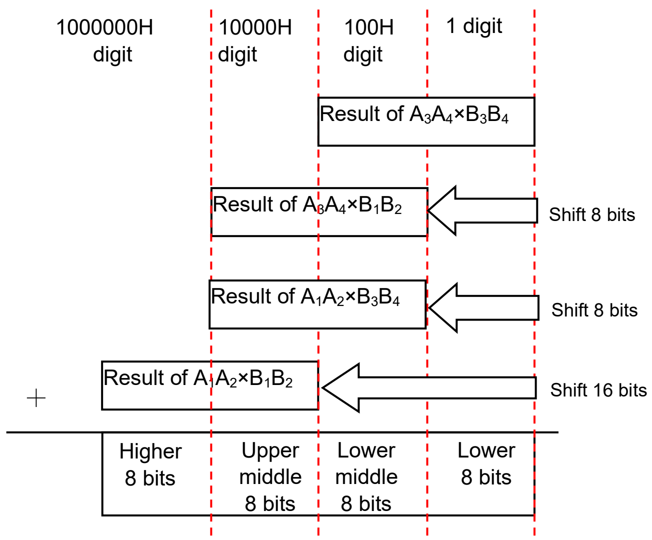

Figure 3.1 Data Scaling of 16 bits×16 bits

3.3 The Way of Thinking of Division

RL78/G10 does not have a division command. Moreover, the multiplier and divider/multiply-accumulator circuit is not carried, either. Therefore, it is necessary to repeat subtraction processing in order to realize division.

The simplest method is making a quotient the number of times that was able to subtract a divisor from a dividend. However, by such a simple method, processing time will be long.

It is common to shorten the processing time by repeating subtraction for the number of the bits of the dividend while shifting a dividend by 1 bit. With division of 16 bits ÷ 16 bits as an example, this method is explained like below.

(1) In order to execute division by repeating subtraction, the work area of the same bit length as a divisor is prepared at higher bits of a dividend and cleared by zero. (Refer to Figure 3.2)

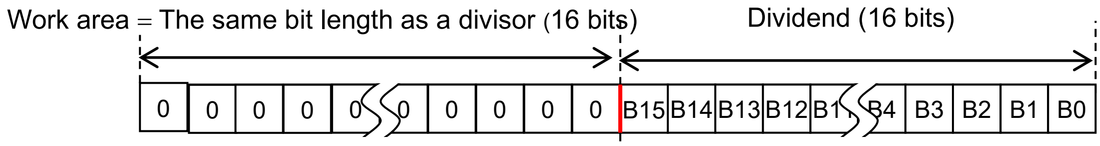

Figure 3.2 Pre-preparation of Calculation

(2) In subtraction processing, a dividend and work area is shifted 1 bit to the left at a time first as the preparation of subtraction. The MSB of dividend is set to the LSB of work area, and the LSB of dividend becomes 0. (Refer to Figure 3.3). The LSB of dividend is used as an area for the quotient in order to obtain the efficiency of processing and domain.

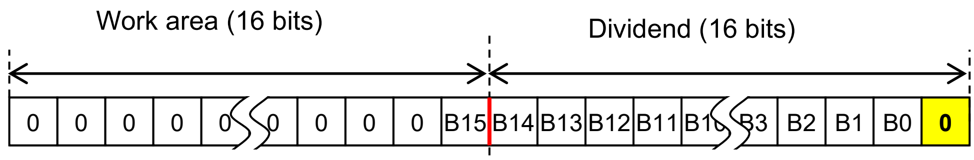

Figure 3.3 Preparation of subtraction

(3) Whether a divisor is able to be subtracted from the work area is confirmed. If subtraction is possible, the quotient of the time becomes 1 and becomes 0 if impossible. This quotient is stored in LSB of a dividend (domain). The remainder at the time of being over is stored in work area. (Refer to **Figure 3.4**)

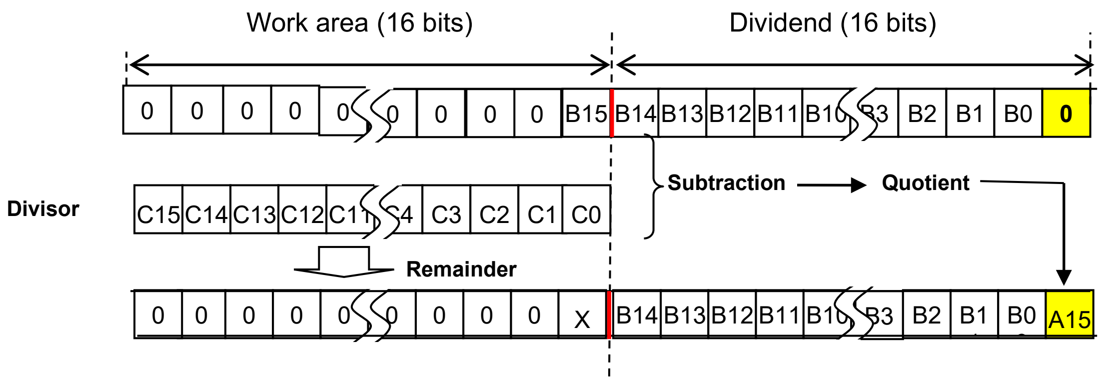

Figure 3.4 Execution of Subtraction

The notes in this calculation are the column numbers of a quotient. After shifting a bit to the left, it subtracts and the quotient is certainly stored in LSB of a dividend area each time. And when both (2) and (3) are performed, the subtraction processing for 1 bit had completed. In the case of Figure 3.4, the quotient bit A15 stored in LSB of a dividend has bit weights of the 15th power of 2. Then, it is stored in the right (lower side) of A15 in order of lighter bit weights by repeating (2) and (3).

(4) The work area and the dividend of a result which performed (2) and (3) for the following digit are shown in Figure 3.5.

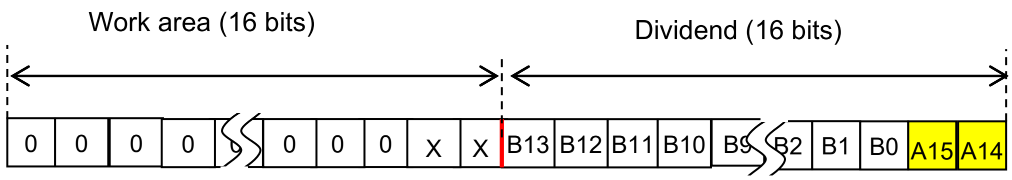

Figure 3.5 The Second Execution Result

(5) The work area and dividend that are results of the 3rd shift and subtraction are shown in Figure 3.6.

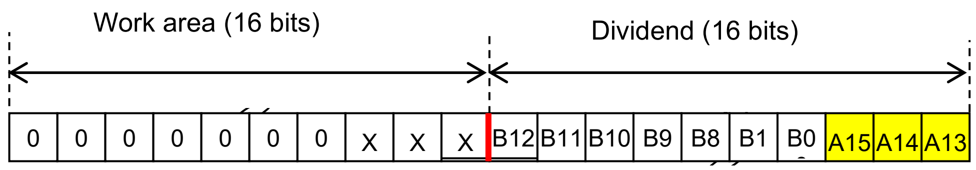

Figure 3.6 The Third Execution Result

(6) The work area and dividend that are results of having repeated the same shift and subtraction 16 times are shown in Figure 3.7. Here, a surplus is stored in work area and a quotient is stored in a dividend.

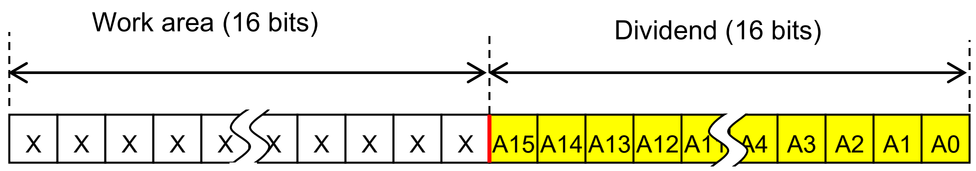

Figure 3.7 The 16th Execution Result

Thus, when a dividend and a quotient share the same area, shift processing can also be shared and a program will be shorter. And since the command to execute decreases, the speed of processing can also be faster.

Although the above example was a case of 16-bit data, the same view is made even if it is 32-bit data.

3.4 Signed Data Handling

A sign is checked first in order to handle signed data for the multiplication and division. In the case of a negative number, after memorizing the sign to a flag, an absolute value is obtained by taking the complement of 2 to data. Arithmetic processing is performed in the obtained absolute value. The combination of the sign of two data is checked, and a result is made into a value with a sign if required.

3.5 List of Constants

Table 3.1 lists the constants for the sample program.

Table 3.1 Constants for the Sample Program

Constant	Setting	Description
DNMULC	**16/8**	The number of bytes of a dividend (16 bits)
DNMUL	**16/8**	The number of bytes of a multiplier (16 bits)
DNRES	**DNMUL + DNMULC**	The number of bytes of the result area (32 bits)
DNdivS	32/**8**	The number of bytes of a divisor (32 bits)
DNdivD	32/**8**	The number of bytes of a dividend (32 bits)
DNQUO	**DNdivD×8**	The number of times of calculation

3.6 List of Variables

Table 3.2 lists variables that are used in this sample program.

Table 3.2 Variables for the Sample Program

Variable Name	Description
RREG0	Multiplicand area/Lower part of divisor area
RREG1	Multiplier area/Upper part of divisor area
RREG2	Multiplication result area/Product-sum addition area/Dividend and quotient area
RREG3	Work and remainder area

3.7 List of Functions (Subroutine)

Table 3.3 lists the functions that are used in this sample program.

Table 3.3 List of Functions (Subroutine)

Functions (Subroutine) Name	Description
M16bitX16bit	Multiplication of 16-bit data (unsigned)
M16bitA32bit	16-bit data is multiplied and the result is added to 32-bit data. (unsigned)
D32bits_32bitS	Division of 32-bit data (unsigned, divisor checked)
D32bit_32bitS2	Division of 32-bit data (unsigned, no divisor check)

3.8 Function (Subroutine) Specifications

This section describes the specifications for the functions (subroutine) that are used in the sample program.

[Function Name] M16bitX16bit

Synopsis	Multiplication of 16-bit data (unsigned)	
Explanation	Multiplies the multiplicand stored in RREG0 by multiplier stored in RREG0, and its result will be stored in RREG2.	
Arguments	• RREG0	Multiplicand (16 bits)
	• RREG1	Multiplier (16 bits)
Return value	• RREG2	• Product (32 bits)
Remarks	Registers to be used	A, X, B, C, D, E, H, L
	Stacks to be used	4+4 bytes

[Function Name] M16bitA32bit

Synopsis	Multiplies 16-bit data by 16-bit data, and the result is added to 32-bit data. (unsigned)	
Explanation	Adds the multiplication result of data stored in RREG0 and RREG1 on RREG2.	
Arguments	• RREG0	Multiplicand (16 bits)
	• RREG1	Multiplier (16 bits)
	• RREG2	Augend (32 bits)
Return value	• RREG2	Product-sum (32 bits)
Remarks	Registers to be used	A, X, B, C, D, E, H, L
	Stacks to be used	4+4 bytes

[Function Name] D32bit_32bitS

Synopsis	Division of 32-bit data (unsigned, divisor checked)	
Explanation	Confirms whether a divisor is 0, and if it is 0, returns in an error. If a divisor is not 0, division is done for the data of RREG2 by the data of RREG0, RREG1, and a quotient will be stored in RREG2, a remainder will be stored in RREG3.	
Arguments	• RREG0	Divisor (lower 16 bits)
	• RREG1	Divisor (higher 16 bits)
	• RREG2	Dividend (32 bits)
Return value	• RREG2	• Quotient (32 bits)
	• RREG3	• Remainder (32 bits)
	• Carry flag	• 0: Normal termination • 1: Error (Divisor is 0.)
Remarks	Registers to be used	A, X, B, C, L
	Stacks to be used	4 bytes (only for CALL command)

[Function Name] D32bit_32bitS2

Synopsis	Division of 32 bits data (unsigned, no divisor check)	
Explanation	Divides the data of RREG2 by the data of RREG0, RREG1. And then a quotient is stored in RREG2, and a remainder is stored in RREG3, and. (The divisor check of D32 bits_32 bits S is omitted.)	
Arguments	• RREG0	Divisor (lower 16 bits)
	• RREG1	Divisor (higher 16 bits)
	• RREG2	Dividend (32 bits)
Return value	• RREG2	• Quotient (32 bits)
	• RREG3	• Remainder (32 bits)
	• Carry flag	• 0: Normal termination
Remarks	Registers to be used	A, X, B, C, L
	Stacks to be used	4 bytes (only for CALL command)

3.9 Flowcharts

3.9.1 Main Processing

Figure 3.8 shows the main processing.

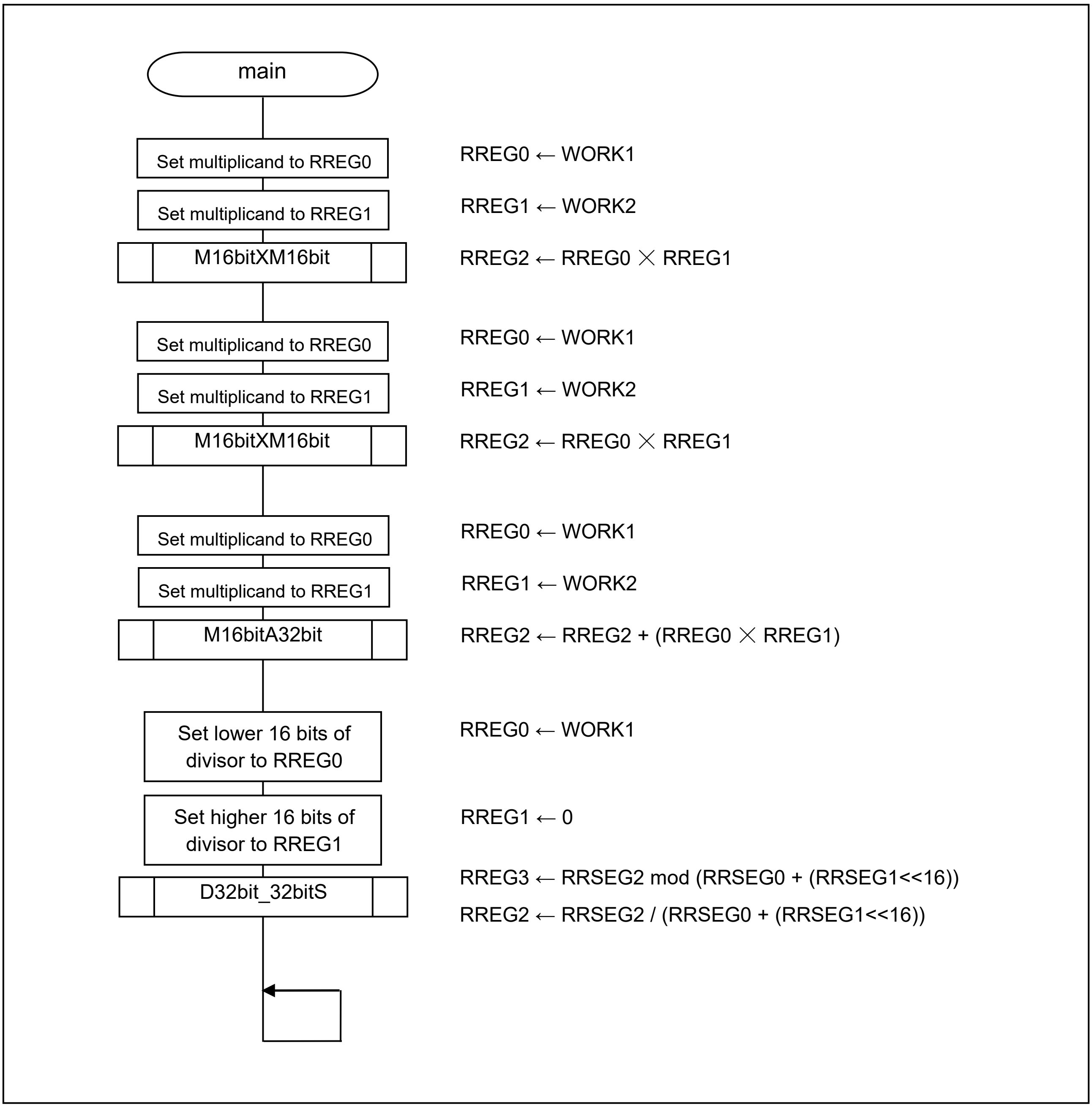

Figure 3.8 Main Processing

3.9.2 Multiplication Processing of 16 Bits Data

Figure 3.9 shows multiplication processing of 16 bits data.

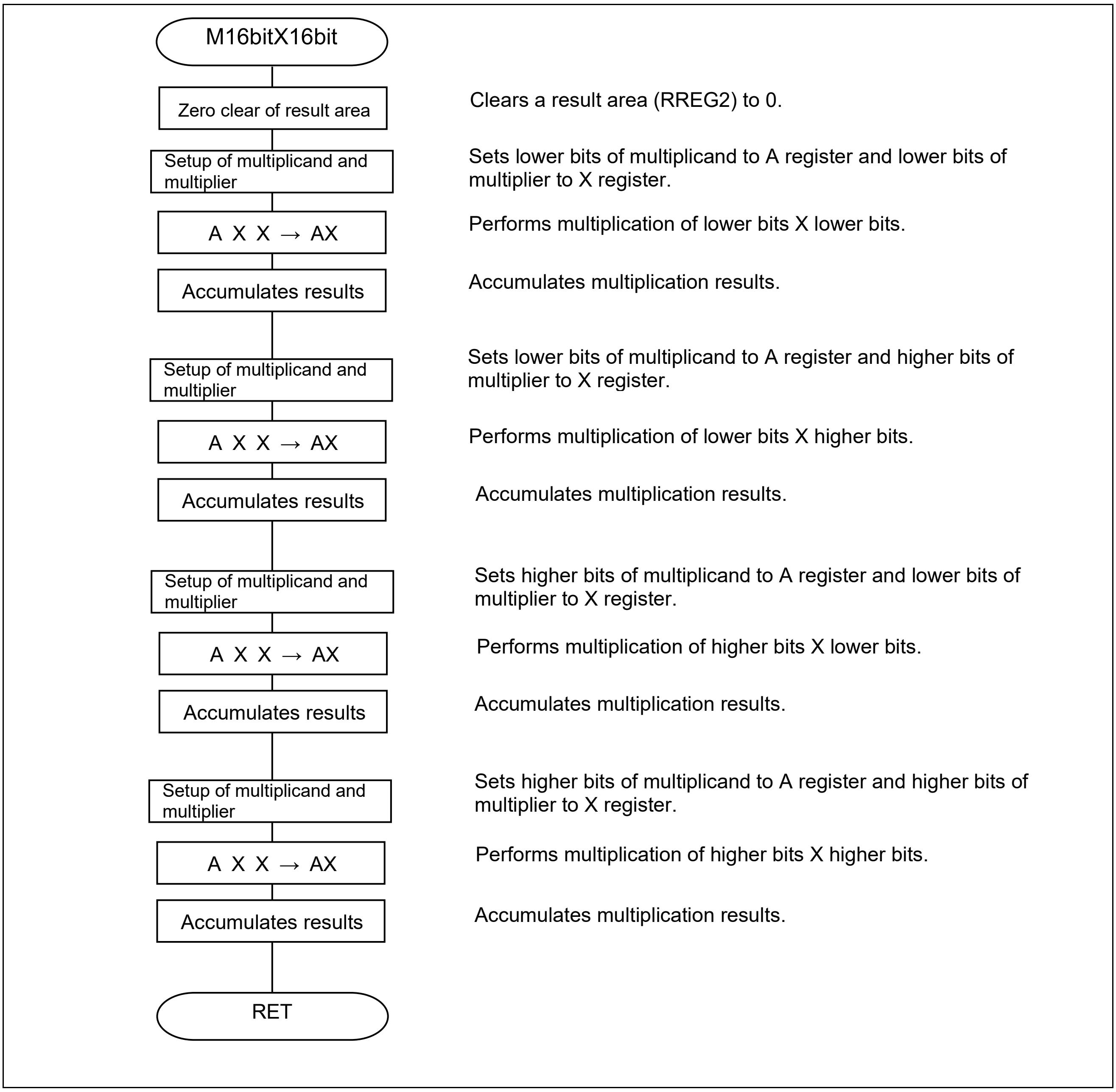

Figure 3.9 Multiplication Processing of 16 Bits Data

3.9.3 Multiply and Accumulation Processing

Figure 3.10 shows multiply and accumulation processing.

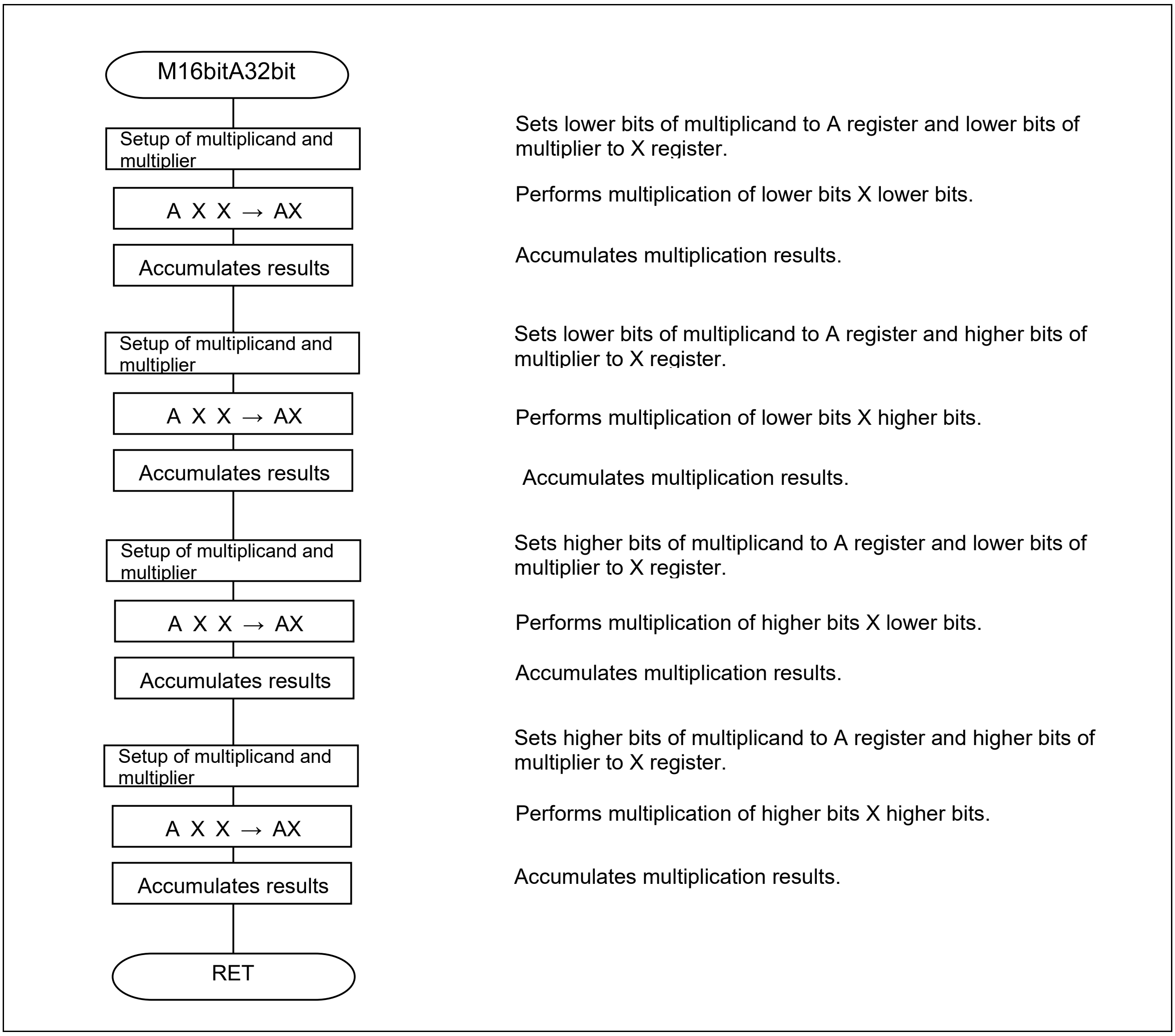

Figure 3.10 Multiply and Accumulation Processing

3.9.4 Division Processing of 32 Bits

Figure 3.11 shows division processing of 32 bits.

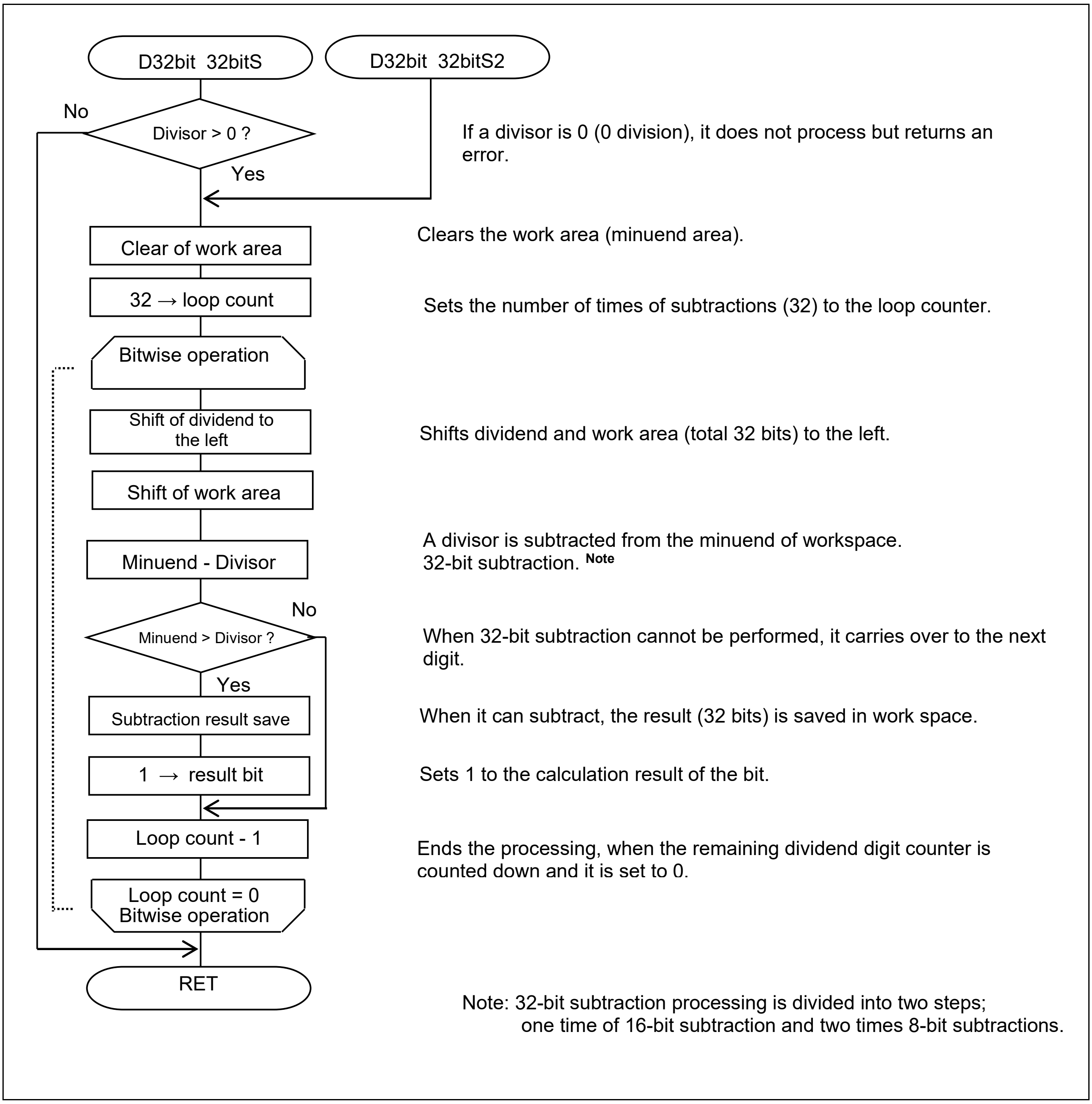

Figure 3.11 Division Processing of 32 Bits

4. Sample Code

The sample code is available on the Renesas Electronics Website.

5. Documents for Reference

RL78/G10 User's Manual: Hardware (R01UH0384E)
RL78 Family User's Manual: Software (R01US0015E)
(The latest versions of the documents are available on the Renesas Electronics Website.)

Technical Updates/Technical Brochures
(The latest versions of the documents are available on the Renesas Electronics Website.)

Revision History: RL78/G10 Multiplication and Division Program CC-RL

Rev.	Date	Description	
		Page	Summary
1.00	Nov. 27, 2015	—	First edition issued
1.01	Oct. 05, 2016	—	Error correction
1.10	June. 24, 2022	3	Operation check condition is updated.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.