RL78/F23, F24

Example of Using Application Accelerator Unit

R01AN6846EJ0100
Rev.1.00
2023. 7.30

Introduction

The Application Accelerator Unit (AAU) of RL78/F23, F24 is the dedicated arithmetic assist hardware to reduce the software load for the algorithm processing on motor control and on DC/DC converter control.

The AAU supports the following algorithmic operations.

Algorithm Mode	Example of Usage	Execution Cycle	Reference
Sine operation	Coordinate transformation for motor vector control, etc.	1 clock	1.1
Cosine operation		1 clock	
Clarke and Park transformation	Used in motor vector control (3-phase <->2-phase conversion, and PI-control).	7 clocks	1.1
Inverse Park (I-Park) transformation		6 clocks	
Inverse Clarke (I-Clarke) transformation		5 clocks	
PI control for the vector control		15 clocks	
I-Park and I-Clarke, and PI control		22 clocks	
I-Park and I-Clarke transformation		11 clocks	
PI control for DC/DC converter control operation (1ch, 2ch, 3ch)	Power supply control using a DC/DC converter, etc.	6 clocks (1ch) 12 clocks (2ch) 18 clocks (3ch)	1.2
32 bits multiply operation (32bits x 32bits = 64bits)	32-bit precision multiply operations such as digital filters	5 clocks	1.3

Contents

1. Example of Using AAU 3
1.1 Example of Motor Control 3
1.2 Example of Power Control 4
1.3 Example of Digital Filter 5
2. Notes on Using AAU 11
3. References 12
Revision History 13

1. Example of Using AAU

As example of using AAU calculation, show "motor control", "power control", and "digital filter".

1.1 Example of Motor Control

An example of using AAU in motor (vector control) is shown in Figure 1-1.

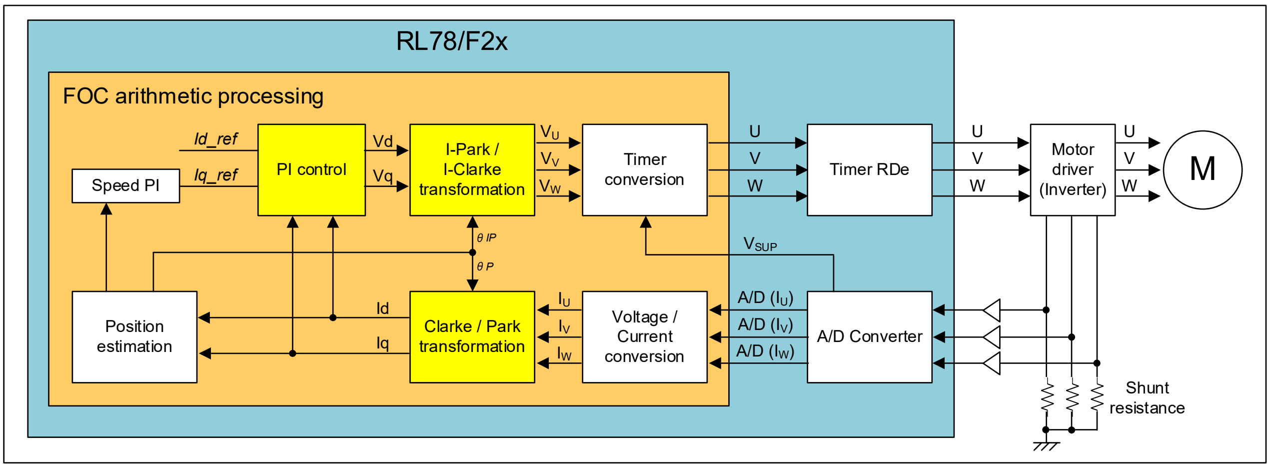

Figure 1-1 Example of Using AAU in Motor (Vector Control)

Table 1-1 AAU Algorithm Mode Used for Motor (Vector Control)

	Algorithm Mode	Mode Operation	Remarks
1	Sine operation	Read Sine value of phase angle (θ).	Used for Clarke / Park transformation of motor (vector control).
2	Cosine operation mode	Read Cosine value of phase angle (θ).	
3	Clarke / Park transformation mode	Executes 3-phase to 2-phase transformation (Clarke transformation) and coordinate axis transformation (Park transformation) in sequence. Note	Clarke transformation: Converts 3-phase current (Iu, Iv, Iw) to 2-phase current (Iα, Iβ). Park transformation: Add phase angle (θ) to fixed coordinates (Iα, Iβ) and convert to rotation coordinates (Id, Iq).
4	I-Park transformation mode	Executes a fixed coordinate transformation (I-Park).	I-Park transformation: Convert from rotating coordinates (Vd, Vq) to fixed coordinates (Vα, Vβ).
5	I-Clarke transformation mode	Executes 2-phase to 3-phase transformation (I-Clarke) Note	I-Clarke transformation: Converts 2-phase voltage (Vα, Vβ) to 3-phase voltage (Vu, Vv, Vw).
6	PI operation for a vector control mode	P control (proportional control) and I control (integral control) are performed in order to bring the motor d-axis (magnetic flux direction) and q-axis (perpendicular to the d-axis) current closer to the target current value.	PI operation: PI control is performed to set the current command values (target values for stable operation) for the d-axis and q-axis of the motor. The command voltage (Vd*, Vq*) is obtained by PI control.
7	Clarke / Park transformation and PI control mode	Sequentially executes Clarke/Park transformation and PI operation for vector control. Note	Refer to the 3. and 6. above.
8	I-Park and I-Clarke transformation mode	I-Park and I-Clarke transform execute sequentially. Note	Refer to the 4. and 5. above.

Note: Absolute conversion mode or relative conversion mode can be selected during Clarke transform or inverse Clarke transform.

For details of this usage example, please refer to the following document.

- RL78/F24 Single-Shunt Sensorless Vector Control for PMSM by MCU
- RL78/F24 Three-Shunt Sensorless Vector Control for PMSM by MCU

1.2 Example of Power Control

An example of using AAU in DC/DC power control for LED is shown in Figure 1-2.

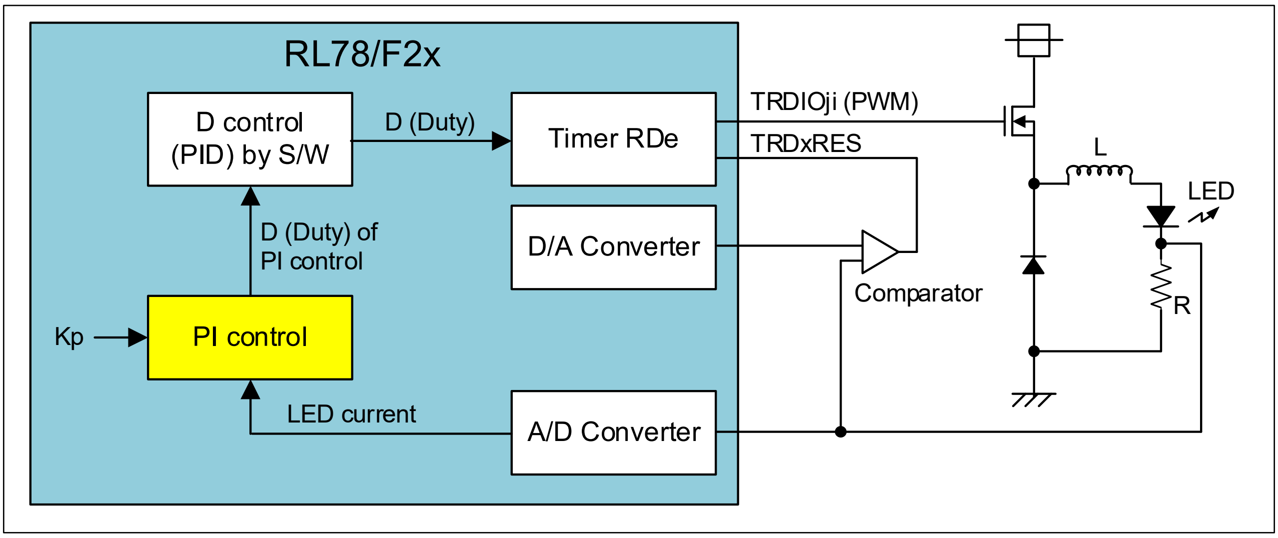

Figure 1-2 Example of Using AAU in Power Control for LED

Table 1-2 AAU Algorithm Mode Used for Power Control of LED

Algorithm Mode	Mode Operation	Remarks
PI control for DC/DC converter control	PI control is performed based on the output voltage that changed before and after sampling, and the switching cycle duty value is calculated to stabilize the target output voltage.	PI control: PI control is performed to supply a stable voltage to the LED by power supply control (switching by PWM) by a DC/DC converter. The duty value (voltage ON time: D(n)) is obtained by PI control. (Figure 1-2 performs D (differential) control by software after PI control)

For usage example (detail) of LED power control, please refer to the application note "RL78/F23, F24 DC/DC Control Using Timer RDe and AAU (R01AN6838)".

1.3 Example of Digital Filter

An example of using AAU in digital filter (IIR filter) is shown in Figure 1-3.

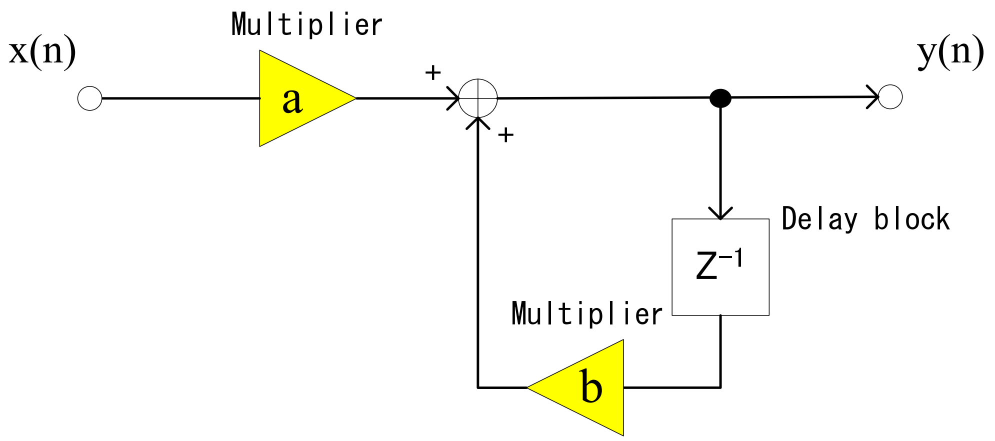

Figure 1-3 Single-Pole IIR Filter Block

Table 1-3 AAU Algorithm Mode Using the Digital Filter (IIR Filter)

Algorithm Mode	Mode Operation	Remarks
32 bits multiply operation	Multiplication of multiplier (signed long: 32 bits) and multiplicand (signed long: 32 bits) are performed.	It is used in arithmetic processing such as multipliers of digital filters.

The single-pole IIR filter shown in the figure is a recursive digital filter that has a feedback loop and the previous output y(n-1) affects the output y(n) due to the delay block. This single-pole IIR filter output is calculated by:

$$\mathrm{y(n)} = ax(n) + by(n-1)$$

"x(n)" is input signal, "a" and "b" are constants representing filter characteristics.

An IIR filter uses audio data that has passed through an A/D converter as an input signal, and is used for signal processing such as cutting high-frequency components with a digital filter.

Figure 1-4 shows the flow chart for implementing the output of a single-pole IIR filter using AAU multiplication mode, and Figure 1-5 shows the source code.

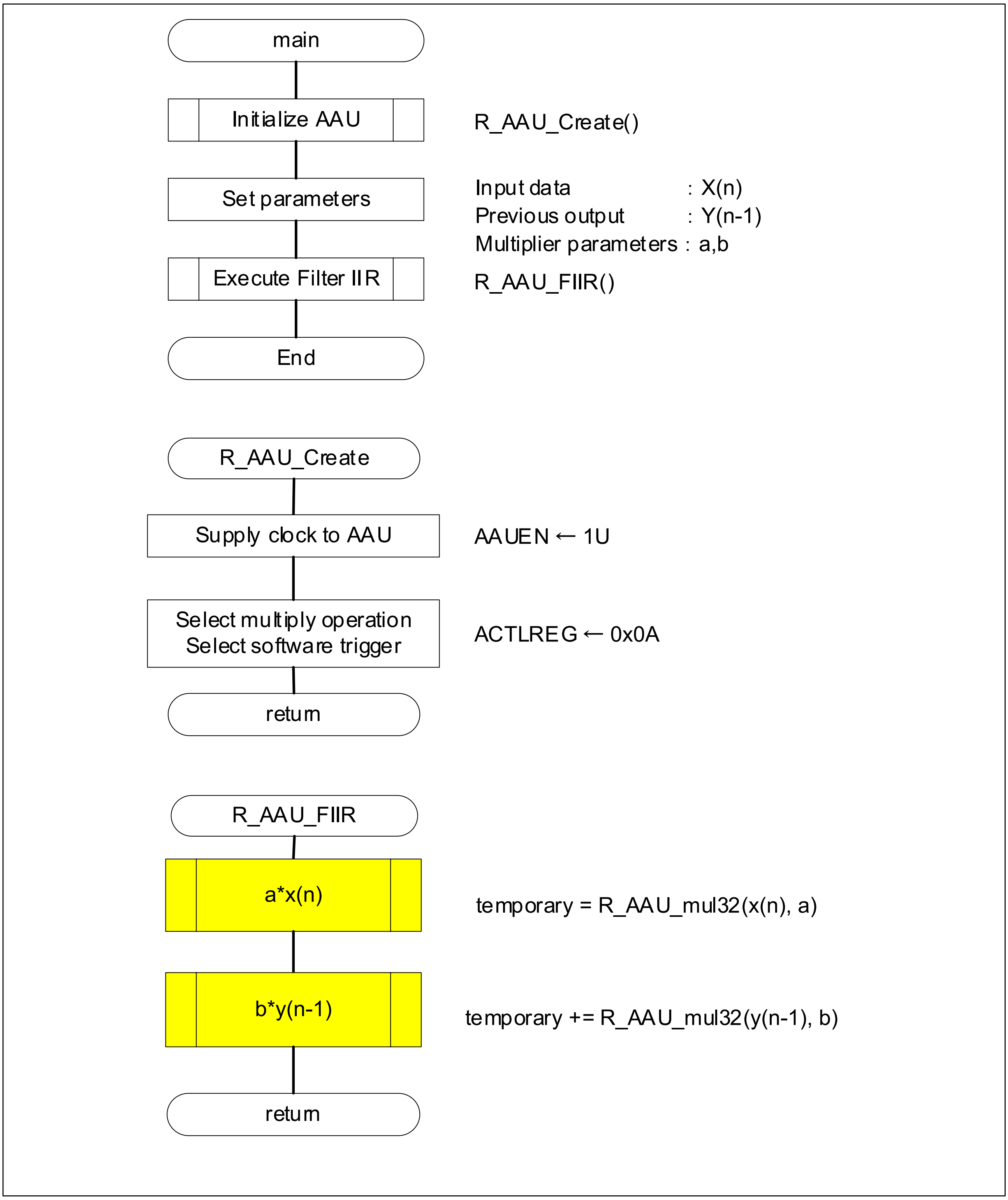

Figure 1-4 Single-Pole IIR Filter Flow (1/2)

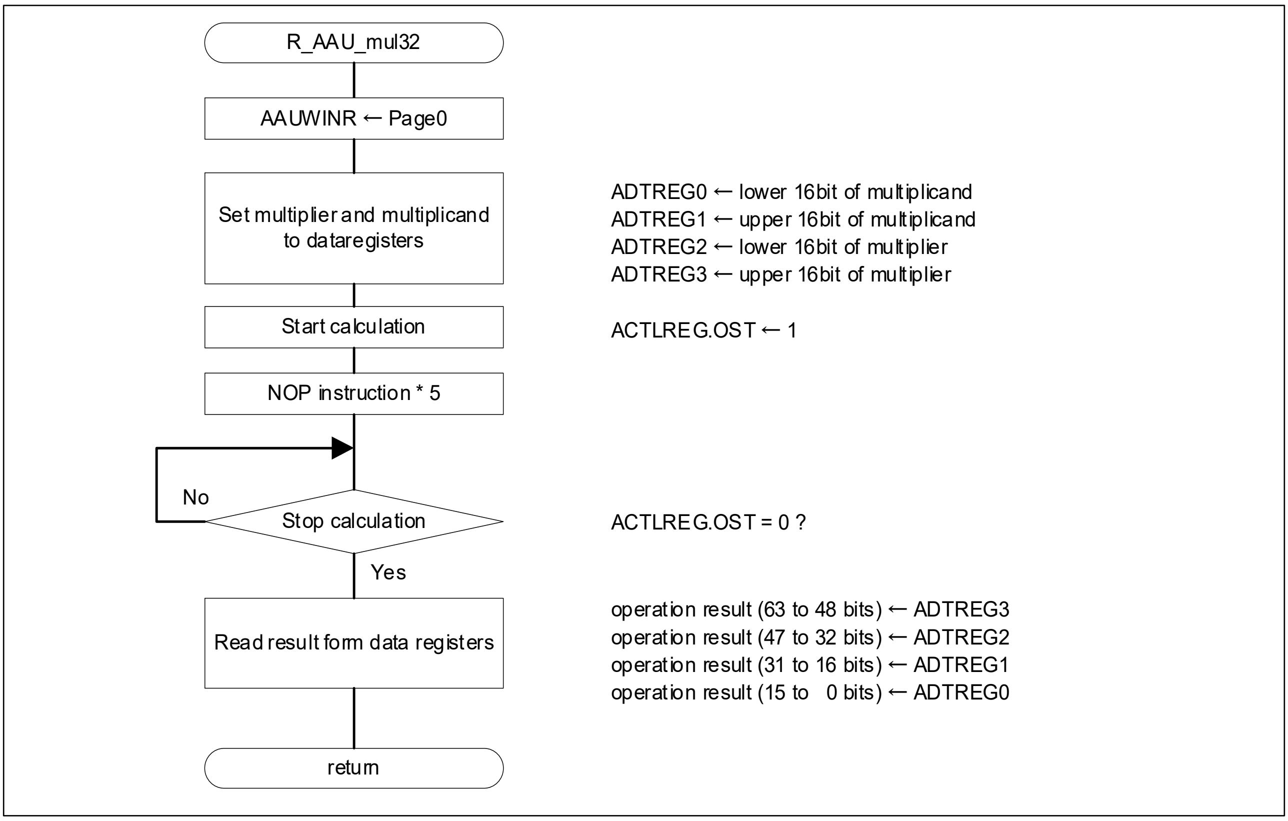

Figure 1-4 Single-Pole IIR Filter Flow (2/2)

```
int64_t R_AAU_mul32(int32_t src, int32_t src2)
{
    int64_t temporary = 0;

    AAUWINR = _00_AAU_WINDOW_PAGE0;

    /* Set calculation data */
    ADTREG0 = (uint16_t) (src  & 0x0000FFFF);
    ADTREG1 = (uint16_t)((src  & 0xFFFF0000) >> 16U);
    ADTREG2 = (uint16_t) (src2 & 0x0000FFFF);
    ADTREG3 = (uint16_t)((src2 & 0xFFFF0000) >> 16U);

    ACTLREG |= _01_AAU_OPERATION_START;

    /* wait for calculation end */
    NOP(); NOP(); NOP(); NOP(); NOP();

    while((ACTLREG & _01_AAU_OPERATION_START) != 0x01);

    temporary = ((int64_t)ADTREG3 << (48U)) | ((int64_t)ADTREG2 << (32U)) |
                ((int64_t)ADTREG1 << (16U)) | (int64_t)ADTREG0;
    return temporary;
}
void R_AAU_Create(void)
{
    AAUEN = 1U;
    /* Set the multiply operation and start trigger */
    AAUWINR = _00_AAU_WINDOW_PAGE0;
    ACTLREG = (_02_AAU_START_TRG_OST | _00_AAU_START_TRG_DATA_SET);
    ACTLREG |=(_08_AAU_OPERATION_MULW | _00_AAU_TRANS_MODE_POWER );
}
void R_AAU_FIIR(int32_t input)
{
    int64_t temporary;

    temporary  = R_AAU_mul32(input, param[0]);          /* ax(n)    */
    temporary += R_AAU_mul32(g_delay, param[1]);        /* by(n-1) */

    g_delay = (int32_t)(temporary & 0x00000000FFFFFFF0);
}
void main(void)
{
    R_AAU_Create();
    /* initial value */
    input = 0x01234567;                 /* x(n)     */
    g_delay = 0x00003333;               /* y(n-1)  */
    param[0] = 5;                       /* a        */
    param[1] = 1;                       /* b        */

    R_AAU_FIIR(input);                  /* y(n)     */
    while(1);
}
```

Figure 1-5 Implementation Example of Digital Filter (IIR Filter)

Multiply operation mode supports multiplication of signed 32bit data (32 bits × 32 bits = 64 bits). When performing multiply operation, the multiplier and multiplicand are used by concatenating two registers as shown in Figure 1-6. **When using negative data, set it in the register in two's complement.**

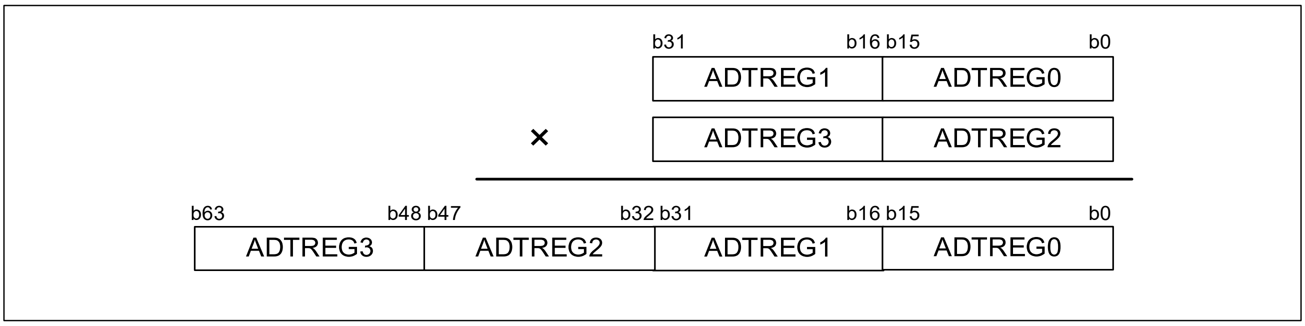

Figure 1-6 Register Used in Multiply Mode (32 bits data)

Some examples of register settings and result when using multiply operation mode is shown below.

(1) "Positive data" × "Positive data"

An example of "01234567H" × "01234567H" is shown in Figure 1-7. When setting multiplier and multiplicand data in registers, the most significant bit (b31) of the two concatenated registers is used as the sign bit, so be careful not to use it for multiplier and multiplicand data.

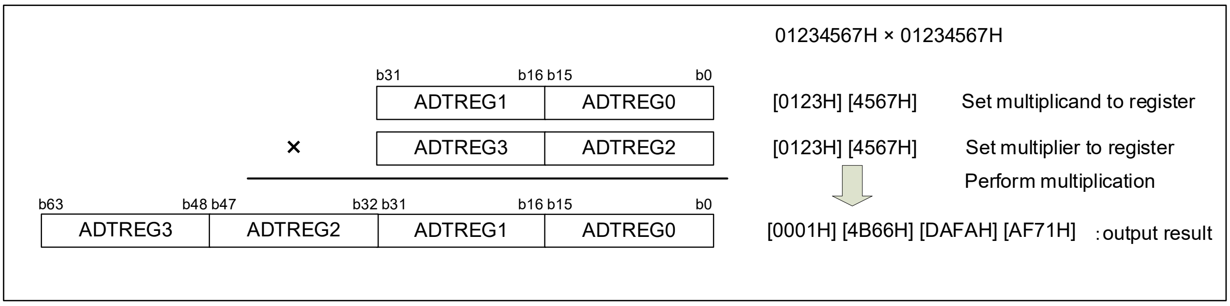

Figure 1-7 Register Setting a "Positive data" × a "Positive data", and Output Result

(2) "Positive data" × "Negative data"

An example of "01234567H" × "-1" is shown in Figure 1-8. If you use a negative data for the multiplier, set the data to the register in two's complement. Also, when an operation with a negative result is performed, the result data is output to the register as two's complement data.

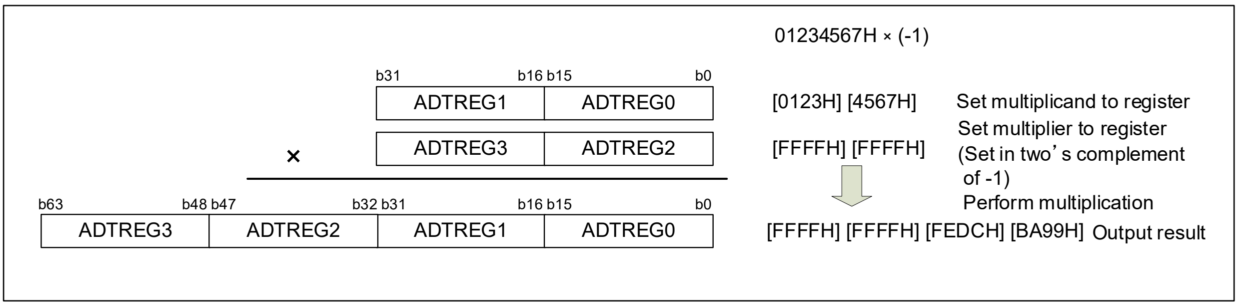

Figure 1-8 Register Setting a "Positive data" × a "Negative data", and Output Result

(3) "Negative data" × "Negative data"

An example of "89ABCDEFH" × "-1" is shown in Figure 1-9. When using a negative data for the multiplicand and multiplier, set the data to the register in two's complement.

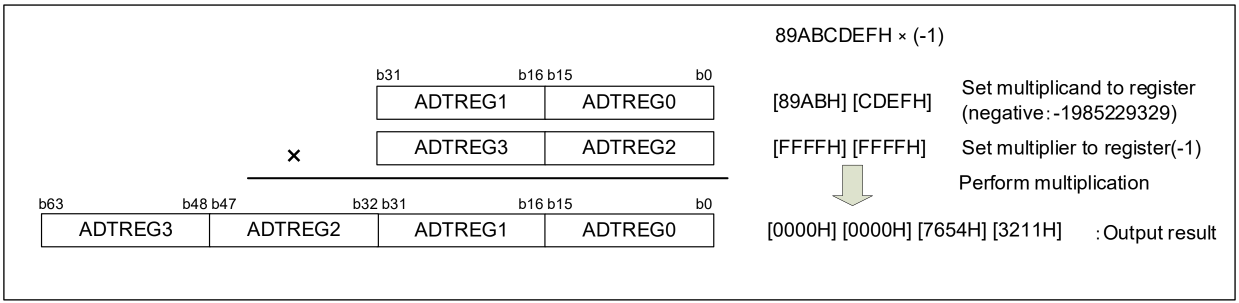

Figure 1-9 Register Setting a "Negative data" × a "Negative data", and Output Result

2. Notes on Using AAU

(1) For each AAU register, select the access page with the AAUPAGE [1:0] bits of the AAUWINR register before accessing the target register.

(2) Do not rewrite the AAU registers while AAU is operating (OST bit of ACTLREG register is "1").

(3) AAU calculation start conditions:

- When the STM bit of the ACTLREG register is "0", the operation is started by writing to the final operand trigger register (see the User's Manual: Hardware for details).
- When the STM bit of the ACTLREG register is "1", the operation is started by writing "1" to the OST bit of the ACTLREG register.

(4) AAU calculation completion determination:

- The software waits for the number of execution cycles of the target operation. Check the execution cycle of each operation in the User's Manual: Hardware (Example: NOP instruction × execution cycle wait)
- Wait until OST bit of ACTLREG register becomes "0".

(5) When using some operation modes, store and restore registers as necessary. The table below shows the registers used in each operation mode.

Table 2-1 Registers Used in Each Operation Mode

Algorithm mode	Input to AAU	Output from AAU
Sine operation	ADTREG0	ADTREG0
Cosine operation	ADTREG0	ADTREG0
32 bits multiply operation	ADTREG0, ADTREG1, ADTREG2, ADTREG3	ADTREG0, ADTREG1, ADTREG2, ADTREG3
Clarke and Park transformation	ADTREG0, ADTREG1, ADTREG2	ADTREG0, ADTREG1
I-Park transformation	ADTREG0, ADTREG1, ADTREG2	ADTREG0, ADTREG1
I-Clarke transformation	ADTREG0, ADTREG1	ADTREG0, ADTREG1, ADTREG2
PI control for motor operation	ADTREG0, ADTREG1 Initial setting: AIDREF, AIQREF, AKPD, AKID, AKPQ, AKIQ, AILIM, APILIM, AKRAG	ADTREG0, ADTREG1 Internal use: AIDBFL, AIDBFH, AIQBFL, AIQBFH, ADOVER, AQOVER
Clarke/Park transformation and PI control for motor operation	ADTREG0, ADTREG1, ADTREG2 Initial setting: AIDREF, AIQREF, AKPD, AKID, AKPQ, AKIQ, AILIM, APILIM, AKRAG	ADTREG0, ADTREG1, ADTREG2, ADTREG3 Internal use: AIDBFL, AIDBFH, AIQBFL, AIQBFH, ADOVER, AQOVER
I-Park/I-Clarke transformation	ADTREG0, ADTREG1, ADTREG2	ADTREG0, ADTREG1, ADTREG2
PI control for DC/DC converter Note	ADTREGn Initial setting: ALnREF, ALnOFS, AKI1, AKI2, ADUTYMX	ADTREGn Internal use: ADUTYLn, AIPLn

Note: The register used differs depending on the algorithm mode of the ACTLREG register.
n = 1 for 1 channel, n = 1, 2 for 2 channels. And for 3 channels, n = 1, 2, 3.

3. References

Documents referenced in this application note are shown below. When referring to these documents, make sure to obtain the latest version of each document from Renesas Electronics website.

- RL78/ F23, F24 User's Manual: Hardware Rev. 1.00
- RL78/F24 Single-Shunt Sensorless Vector Control for PMSM by MCU Rev 1.10
- RL78/F24 Three-Shunt Sensorless Vector Control for PMSM by MCU Rev.1.10
- RL78/F23, F24 DC/DC Control Using the Timer RDe and the AAU Rev.1.00

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	2023.7.30	-	First edition issued

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

Trademarks

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/.