To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

H8/300L SLP Series

Using the Asynchronous Event Counter for Control

Introduction

Using an asynchronous event counter, a series of operation is repeated periodically: a transition from subactive mode to high-speed active mode, reversal of the port output in high-speed active mode, and a transition back to subactive mode.

Target Device

H8/38024

Contents

1. Specifications

1. Using an asynchronous event counter (AEC), a series of operation is repeated once every 524.288 ms: a transition from subactive mode to high-speed active mode, reversal of the port output in high-speed active mode, and a transition back to subactive mode.
2. The 2-MHz event input is applied to the asynchronous event input L pin (AEVL).
3. In this sample task, the AEC is used as a 16-bit asynchronous event counter.

2. Description of Functions

1. In this sample task, an asynchronous event counter (AEC) is used to induce transitions between subactive and active modes and to invert the port output. The features of the AEC are as follows.
   - External events input asynchronously are counted without regard to the basic clock operation.
   - The counter has a 16-bit configuration, and can count up to 65,536 events.
   - The circuit can also be used as two independent 8-bit event counters on different channels.
   - The counter can be reset and its counting-up operation can be halted under software control.
   - Event counter overflow can be detected to automatically generate an interrupt.
   - When not in use, the AEC alone can be placed in a standby state by the module standby mode.
2. Figure 2.1 is a block diagram of the 16-bit asynchronous event counter used in this sample task.

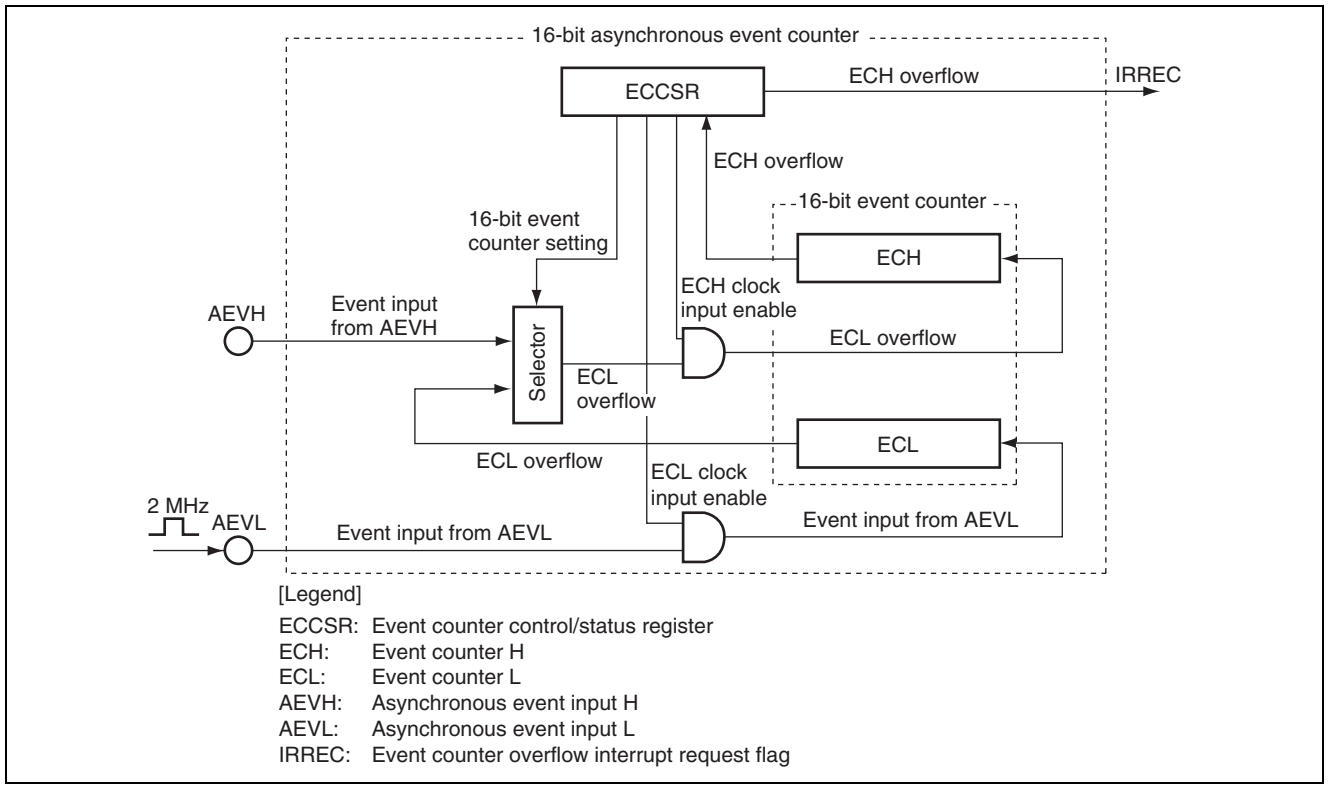


Figure 2.1 Block Diagram of Asynchronous Event Counter

3. Functions of the 16-bit asynchronous event counter are described in table 2.1 below.

Table 2.1 Functions of the 16-bit Asynchronous Event Counter

Register	Function
Input pin edge selection register (AEGSR)	AEGSR is an 8-bit read/write register which selects the edge sensing of rising edge, falling edge, or both edges for AEVH, AEVL and IRQAEC pins.
Event counter control register (ECCR)	ECCR is an 8-bit read-only up-counter which controls the input clock of the counter and IRQAEC/IECPWM
Event counter control/ status register (ECCSR)	ECCSR is an 8-bit read/write register which is used to control counter overflow detection, the counter resetting, and halting of counting-up function. Upon a reset, ECCSR is initialized to H'00.
Event counter H (ECH)	ECH is an 8-bit read-only up-counter which operates either as an independent 8-bit event counter or, in combination with ECL, as the counter for the upper eight bits of a 16-bit event counter. As the input clock signal, either the external asynchronous event AEVH pin, or the overflow signal from the lower 8-bit counter ECH can be selected by the CH2 bit in ECCSR. ECH can be cleared to H'00 by software. Upon a reset, ECH is initialized to H'00.
Event counter L (ECL)	ECL is an 8-bit read-only up-counter which operates either as an independent 8-bit event counter or, in combination with ECH, as the counter for the lower eight bits of a 16-bit event counter. As the input clock signal, the event clock from the external asynchronous event AEVL pin is used. ECL can be cleared to H'00 by software. Upon a reset, ECL is initialized to H'00.
Asynchronous event input H (AEVH)	AEVH is the event input pin for input to event counter H (ECH).
Asynchronous event input L (AEVL)	AEVL is the event input pin for input to event counter L (ECL).
Asynchronous event counter interrupt request flag (IRREC)	When an asynchronous event counter interrupt request is generated, IRREC is set to 1. It is not automatically cleared even after the interrupt is accepted. To clear IRREC, write 0 by software.
Asynchronous event counter interrupt enable (IENEC)	IENEC enables or disables asynchronous event counter interrupt requests.

4. Figure 2.2 shows an example of settings when using the AEC as a 16-bit asynchronous event counter.

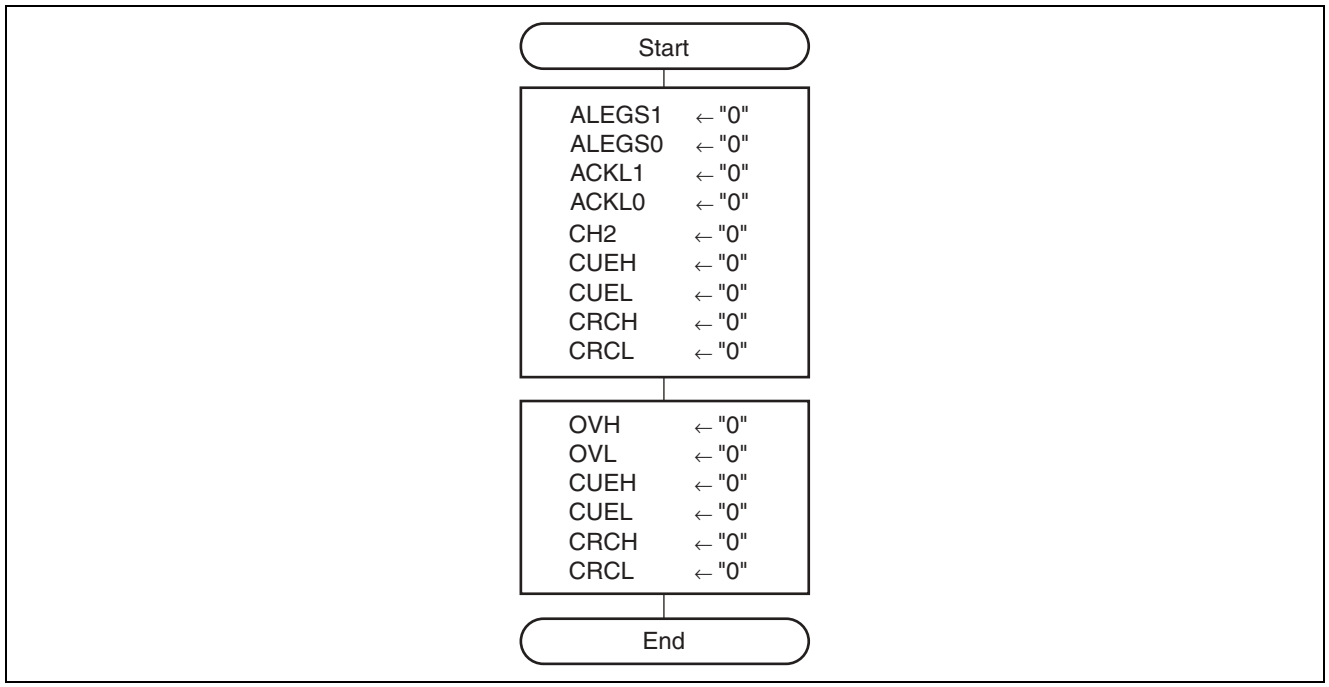


Figure 2.2 Example of Settings for 16-bit Asynchronous Event Counter

Since CH2 is cleared to 0 upon a reset, ECH and ECL operate as a 16-bit event counter after a reset. The AEC also operate as a 16-bit event counter by the settings shown in figure 2.2. The operating clock source is the asynchronous event input from the AEVL pin. When the next clock pulse is input after the count values for both ECH and ECL reach H'FF, ECH and ECL overflow, the OVH flag in ECCSR is set to 1, the count values of ECH and ECL are both reset to H'00, and counting-up is restarted. Upon occurrence of overflow, the IRREC bit in IRR2 is set to 1. At this time, if the IENEC bit in IENR2 is 1, an interrupt request is issued to the CPU.

5. Asynchronous event counter operating modes are shown in table 2.2.

Table 2.2 Asynchronous Event Counter Operating Modes

Operating Mode	Reset	Active	Sleep	Watch	Subactive	Subsleep	Standby	Module Standby
AEGSR	Reset	Functions	Functions	Held*[1]	Functions	Functions	Held*[1]	Held
ECCR	Reset	Functions	Functions	Held*[1]	Functions	Functions	Held*[1]	Held
ECCSR	Reset	Functions	Functions	Held*[1]	Functions	Functions	Held*[1]	Held
ECH	Reset	Functions	Functions	Functions*[1,2]	Functions*[2]	Functions*[2]	Functions*[1,2]	Halted
ECL	Reset	Functions	Functions	Functions*[1,2]	Functions*[2]	Functions*[2]	Functions*[1,2]	Halted

Notes: 1. When an asynchronous external event signal is input, the counter is incremented, but the count overflow H/L flags are not affected.

2 In these modes, ECH and ECL operate when “asynchronous external event” is selected, otherwise the AEC is halted and held pending.

6. Notes on the 16-bit asynchronous event counter
   A. Before reading the values of ECH and ECL, the CUEH and CUEL bits in ECCSR must be cleared to 0 to prevent asynchronous event input to the counter. If the event counter is incremented while being read, the correct value cannot be read.
   B. Use a clock with a frequency of up to 16 MHz for input to the AEVH and AEVL pins, and ensure that the high- and low-level widths of the clock are at least 30 ns. There is no constraint on the duty cycle.
   C. In the case of the AEC used in 16-bit mode, before the clock is input to it, ECCSR setting should be made such that the CUEH bit is first set to 1 and then the CRCH bit to 1 or both CUEH and CRCH are set simultaneously. Thereafter, do not change the value of CUEH during AEC operation in 16-bit mode. If CUEH is changed while in 16-bit mode, ECH will not be incremented correctly.
   D. When ECPWME in AEGSR is 1, event counter PWM is operating and therefore ECPWCRH, ECPWCRL, ECPWDRH, ECPWDRL should not be modified. When changing the data in them, event counter PWM must be stopped by clearing ECPWME to 0 before writing to these registers.
   E The value of Event Counter PWM Data Register and the value of Event Counter PWM Compare Register must be set so that Event Counter PWM Data Register < Event Counter PWM Compare Register. If the settings do not satisfy this condition, do not set ECPWME in AEGSR to 1.
   F. As synchronization is established internally when an IRQAEC interrupt is generated, a maximum error of 1 tcyc will arise between clock halting and interrupt acceptance.
   G. Table 2.3 shows operating modes and event input frequencies.

Table 2.3 Relation between Operating Modes and AEVH/AEVL Pin Event Input Frequencies

Mode		Maximum AEVH/AEVL Pin Input Clock Frequency
Active (high-speed), Sleep (high-speed)		16 MHz
Medium-speed active, Sleep (medium-speed)	(ϕ/16)	$2 \cdot f_{osc}$
	(ϕ/32)	f_{osc}
	(ϕ/64)	$1/2 \cdot f_{osc}$
f_{osc} = 1 MHz to 4 MHz	(ϕ/128)	$1/4 \cdot f_{osc}$
Watch, Subactive, Subsleep,	(ϕ_w/2)	1000 KHz
Standby	(ϕ_w/4)	500 KHz
ϕ_w = 32.768 kHz or 38.4 kHz	(ϕ_w/8)	250 KHz

7. Table 2.4 shows the assignment of functions in this sample task.

Table 2.4 Function Assignment

Function	Assignment
AEGSR	Selects the edge sensing of rising, falling, or both edges for AEVL.
ECCR	Controls AEVL counter input clock.
ECCSR	Sets 16-bit asynchronous event counter function, detects counter overflow, enables/disables the event clock input to ECH and ECL.
ECH	Functions as the upper 8-bit up-counter of a 16-bit event counter, taking the ECL overflow signal as the input clock.
ECL	Functions as the lower 8-bit up-counter of a 16-bit event counter, taking the external asynchronous event input on the AEVL pin as the input clock.
COM	Sets P40/SCK32 pin function.
SCR3	Sets P40/SCK32 pin function.
TMA3	Used to set transition to subactive mode.
AEVL	Functions as the input pin for 2-MHz external asynchronous event input.
P40	Inverts P40 output, triggered by 10 times of ECH and ECL overflows.
PCR40	Sets P40 pin to output pin function.
SYSCR1	Controls power down modes.
SYSCR2	Controls power down modes.
IENDT	Enables/disables direct transition interrupt requests
IENEC	Enables/disables asynchronous event counter interrupt requests.
IRRDT	Indicates whether or not a direct transition interrupt has been requested
IRREC	Indicates whether or not an asynchronous event counter interrupt has been requested.

3. Principle of Operation

1. Figure 3.1 illustrates the operation of this sample task. Asynchronous event counter operation is implemented through hardware and software processing as shown in the figure.

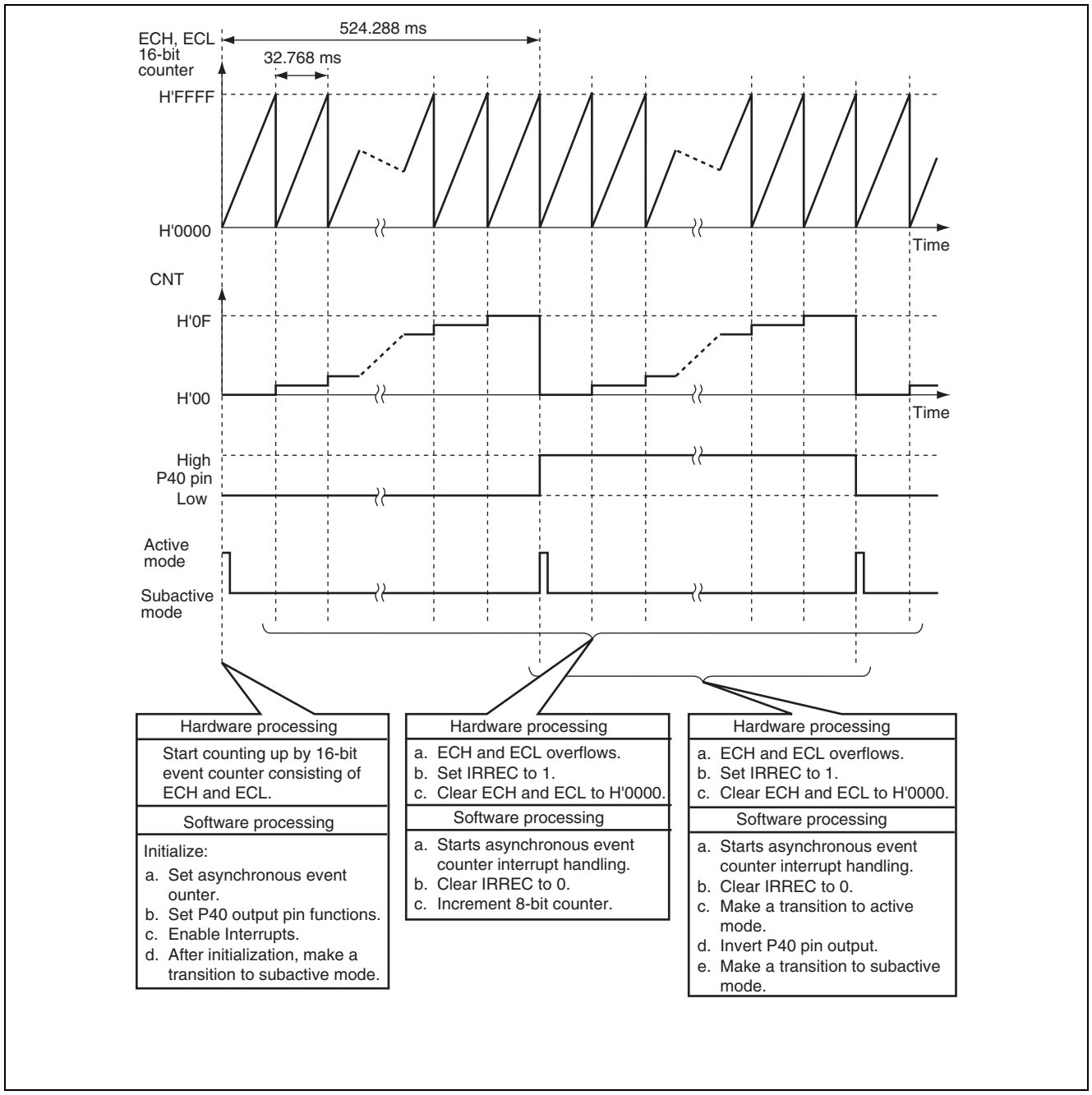


Figure 3.1 Operation Principle of Asynchronous Event Counter

4. Description of Software

4.1 Modules

The modules used in this sample task are shown in table 4.1.

Table 4.1 Description of Modules

Module	Label	Function
Main routine	main	Initializes RAM area for use, port 40, asynchronous event counter and system control register; enables interrupts; executes direct transition to subactive mode; after 524.288 ms, controls port output and executes direct transitions to high-speed active mode or subactive mode.
Asynchronous event counter interrupt handling routine	aecint	An asynchronous event counter interrupt handling routine which clears an interrupt request flag, increments and initializes an 8-bit counter, and after 524.288 ms, sets the flag in RAM.
Direct transition interrupt handling routine	dtint	A direct transition interrupt handling routine which clears the interrupt request flag.

4.2 Arguments

In this sample task, no arguments are used.

4.3 Internal Registers

Table 4.2 shows the internal registers used in this sample task.

Table 4.2 Description of Internal Registers

Register		Function	Address	Setting
AEGSR	ALEGS1 ALEGS0	Input Pin Edge Selection Register(AEC Edge Select L) If ALEGS1 = 0 and ALEGS0 = 0, falling edge of AEVL pin input is detected.	H'FF92 Bit 5 Bit 4	ALEGS1 = 0 ALEGS0 = 0
ECCR	ACKL1 ACKL0	Event Counter Control Register (AEC Clock Select L) If ACKL1 = 0 and ACKL0 = 0, the clock used by ECL is input from AEVL pin.	H'FF94 Bit 5 Bit 4	ACKL1 = 0 ACKL0 = 0
ECCSR	OVH	Event Counter Control/Status Register (Counter Overflow H) A status flag indicating overflow of ECH. If OVH = 0, indicates no overflow of ECH. If OVH = 1, indicates ECH overflow.	H'FF95 Bit 7	0
	OVL	Event Counter Control/Status Register (Counter Overflow L) A status flag indicating overflow of ECL. If OVL = 0, indicates no overflow of ECL. If OVL = 1, indicates ECL overflow.	H'FF95 Bit 6	0

Register		Function	Address	Setting
ECCSR	CH2	Event Counter Control/Status Register (Channel Selection) Selects whether to use ECH and ECL as a single-channel 16-bit event counter, or as independent 8-bit event counters on two channels. If CH2 = 0, ECH and ECL function as a single concatenated 16-bit event counter. If CH2 = 1, ECH and ECL function as independent 8-bit event counters on two channels.	H'FF95 Bit 4	0
	CUEH	Event Counter Control/Status Register (Count-Up Enable H) Enables or disables the event clock input to ECH. If CUEH = 0, disables the event clock input to ECH. If CUEH = 1, enables the event clock input to ECH.	H'FF95 Bit 3	0
	CUEL	Event Counter Control/Status Register (Count-Up Enable L) Enables or disables the event clock input to ECL. If CUEL = 0, disables the event clock input to ECL. If CUEL = 1, enables the event clock input to ECL.	H'FF95 Bit 2	0
	CRCH	Event Counter Control/Status Register (Counter Reset Control H) Controls ECH reset. If CRCH = 0, ECH is reset. If CRCH = 1, ECH reset is cancelled and count-up function is enabled.	H'FF95 Bit 1	0
	CRCL	Event Counter Control/Status Register (Counter Reset Control L) Controls ECL reset. If CRCL = 0, ECL is reset. If CRCL = 1, ECL reset is cancelled and count-up function is enabled.	H'FF95 Bit 0	0
ECH		Event Counter H An 8-bit read-only up-counter; in combination with ECL, it operates as the upper 8 bits of a 16-bit event counter.	H'FF96	H'00
ECL		Event Counter L An 8-bit read-only up-counter; in combination with ECH, it operates as the lower 8 bits of a 16-bit event counter.	H'FF97	H'00
SMR	COM	Serial mode Register (Communication Mode) If COM = 0, P40/SCK32 pin functions as the P40 pin. If COM = 1, P40/SCK32 pin functions as the SCK32 output pin.	H'FFA8 Bit 7	0
SCR3	CKE1 CKE0	Serial Control Register 3 (Clock Enable1,0) If CKE1 = 0, CKE0 = 0, COM = 0 and PCR40 = 0, P40/SCK32 pin functions as the P40 input pin.	H'FFAA Bit 1 Bit 0	CKE1 = 0 CKE0 = 0
TMA	TMA3	Timer Mode Register A (Internal Clock Selector 3) Selects the clock source input to TCA. If TMA3 = 0, PSS is selected as the TCA input clock source, and an interval timer function is selected for timer A If TMA3 = 1, PSW is selected as the TCA input clock source, and a clock time base function is selected for timer A	H'FFB0 Bit 3	1

Register		Function	Address	Setting
PMR3	AEVL	Port Mode Register 3(P37/AEVL Pin Function Switch) Sets the P37/AEVL pin function. If AEVL = 0, the P37/AEVL pin functions as the P37 input/output pin If AEVL = 1, the P37/AEVL pin functions as the AEVL input pin	H'FFCA Bit 7	1
PDR4	P40	Port Data Register 4 (P40) Stores the P40 pin data. If P40 = 0, the P40 pin output level is low If P40 = 1, the P40 pin output level is high	H'FFD7 Bit 0	0
PCR4	PCR40	Port Control Register 4 (Port Control Register 40) Controls the P40 pin input/output. If PCR40 = 0, the P40 pin functions as an input pin If PCR40 = 1, the P40 pin functions as an output pin	H'FFE7 Bit 0	1
SYSCR1	SSBY	System Control Register 1 (Software Standby) Specifies transition to standby mode or watch mode. If SSBY = 0, after a SLEEP instruction is executed in active mode, a transition is made to sleep mode, or after a SLEEP instruction is executed in subactive mode, a transition is made to subsleep mode If SSBY = 1, after a SLEEP instruction is executed in active mode, a transition is made to standby mode or to watch mode, or after a SLEEP instruction is executed in subactive mode, a transition is made to watch mode	H'FFF0 Bit 7	1
	STS2 STS1 STS0	System Control Register 1 (Standby Timer Select 2 to 0) Specifies the time for the CPU and peripheral functions to wait until the clock stabilizes when, triggered by a specific interrupt, the standby mode or watch mode is terminated and a transition is made to active mode. When STS2 to STS1 = 000, standby time is 8,192 states When STS2 to STS1 = 001, standby time is 16,384 states When STS2 to STS1 = 010, standby time is 1,024 states When STS2 to STS1 = 011, standby time is 2,048 states When STS2 to STS1 = 100, standby time is 4,096 states When STS2 to STS1 = 101, standby time is 2 states When STS2 to STS1 = 110, standby time is 8 states When STS2 to STS1 = 111, standby time is 16 states	H'FFF0 Bit 6 Bit 5 Bit 4	STS2 = 0 STS1 = 0 STS0 = 0
	LSON	System Control Register 1 (Low Speed On Flag) When watch mode is terminated, selects either the system clock (ϕ) or the subclock (ϕ_{sub}) as the CPU operating clock. If LSON = 0, selects the system clock (ϕ) as the CPU operating clock If LSON = 1, selects the subclock (ϕ_{sub}) as the CPU operating clock	H'FFF0 Bit 3	1

Register		Function	Address	Setting
SYSCR2	NESEL	System Control Register 2 (Noise Elimination Sampling Frequency Selection) Selects the frequency at which the watch clock signal (ϕ_w) output by the subclock pulse generator is sampled using the oscillator clock (ϕ_{osc}) output by the system clock pulse generator. If NESEL = 0, sampling frequency is ϕ_{osc} /16. If NESEL = 1, sampling frequency is ϕ_{osc} /4.	H'FFF1 Bit 4	1
	DTON	System Control Register 2 (Direct Transfer On Flag) Specifies whether or not to make direct transitions among high-speed active mode, medium-speed active mode, and subactive mode when a SLEEP instruction is executed. When DTON = 0, if a SLEEP instruction is executed in active mode, a transition is made to standby mode, watch mode or sleep mode; if a SLEEP instruction is executed in subactive mode, a transition is made to watch mode or subsleep mode. When DTON = 1, if a SLEEP instruction is executed in high-speed active mode, a direct transition is made to medium-speed active mode (when SSBY = 1, MSON = 1, LSON = 0) or to subactive mode (when SSBY = 1, TMA3 = 1, LSON = 1); if a SLEEP instruction is executed in medium-speed active mode, a direct transition is made to high-speed active mode (when SSBY = 0, MSON = 0, LSON = 0) or to subactive mode (when SSBY = 1, TMA3 = 1, LSON = 1); and if a SLEEP instruction is executed in subactive mode, a direct transition is made to high-speed active mode (when SSBY = 1, TMA3 = 1, LSON = 0, MSON = 0) or to medium-speed active mode (when SSBY = 1, TMA3 = 1, LSON = 0, MSON = 1).	H'FFF1 Bit 3	1
	MSON	System Control Register 2 (Medium Speed On Flag) Selects whether to operate in high-speed active mode or in medium-speed active mode after the standby mode, watch mode, or sleep mode is terminated,. If MSON = 0, operates in high-speed active mode. If MSON = 1, operates in medium-speed active mode.	H'FFF1 Bit 2	0
	SA1 SA0	System Control Register 2 (Subactive Mode Clock Select 1, 0) Selects the CPU operating clock (ϕ_w/8, ϕ_w/4, ϕ_w/2) in subactive mode. If SA1 = 0 and SA0 = 0, ϕ_w/8 is selected If SA1 = 0 and SA0 = 1, ϕ_w/4 is selected If SA1 = 1 and SA0 = *, ϕ_w/2 is selected Note: * Don't care	H'FFF1 Bit 1 Bit 0	SA1 = 0 SA0 = 0

Register		Function	Address	Setting
IRR2	IRRDT	Interrupt Request Register 2 (Direct Transition Interrupt Request Flag) Indicates whether there is any direct transition interrupt requested. If IRRDT = 0, indicates that no direct transition interrupt has been requested. If IRRDT = 1, indicates that a direct transition interrupt has been requested.	H'FFF7 Bit 7	0
	IRREC	Interrupt Request Register 2 (Asynchronous Event Counter Interrupt Request Flag) Indicates whether there is any asynchronous event counter interrupt requested. If IRREC = 0, indicates that no asynchronous event counter interrupt has been requested. If IRREC = 1, indicates that an asynchronous event counter interrupt has been requested.	H'FFF7 Bit 0	0
IENR2	IENDT	Interrupt Enable Register 2 (Direct Transition Interrupt Enable) Enables or disables direct transition interrupt requests. If IENDT = 0, disables direct transition interrupt requests. If IENDT = 1, enables direct transition interrupt requests.	H'FFF4 Bit 7	1
	IENEC	Interrupt Enable Register 2 (Asynchronous Event Counter Interrupt Enable) Enables or disables asynchronous event counter interrupt requests. If IENEC = 0, disables asynchronous event counter interrupt requests If IENEC = 1, enables asynchronous event counter interrupt request	H'FFF4 Bit 0	1

4.4 Description of RAM

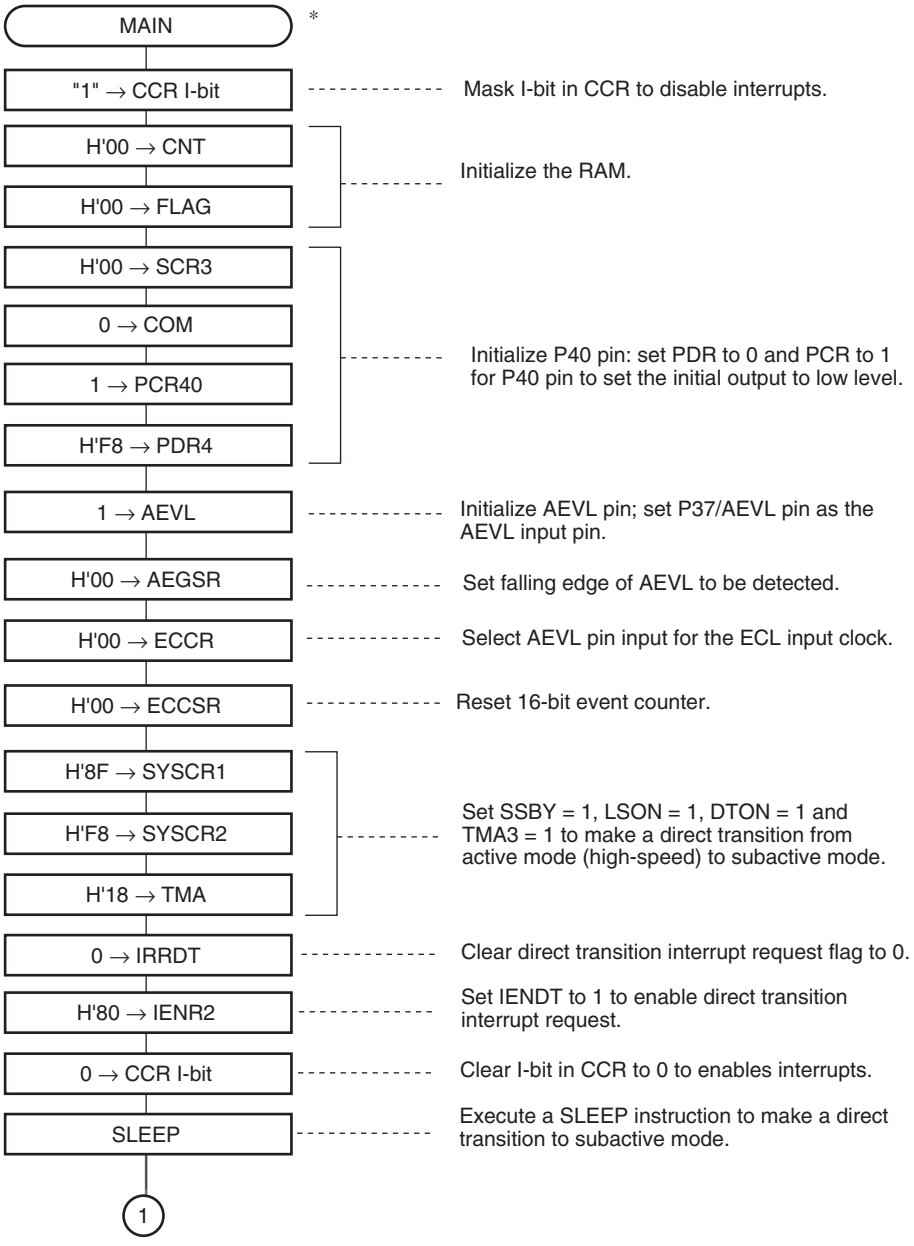
Table 4.3 describes the RAM area used in this sample task.

Table 4.3 Description of RAM

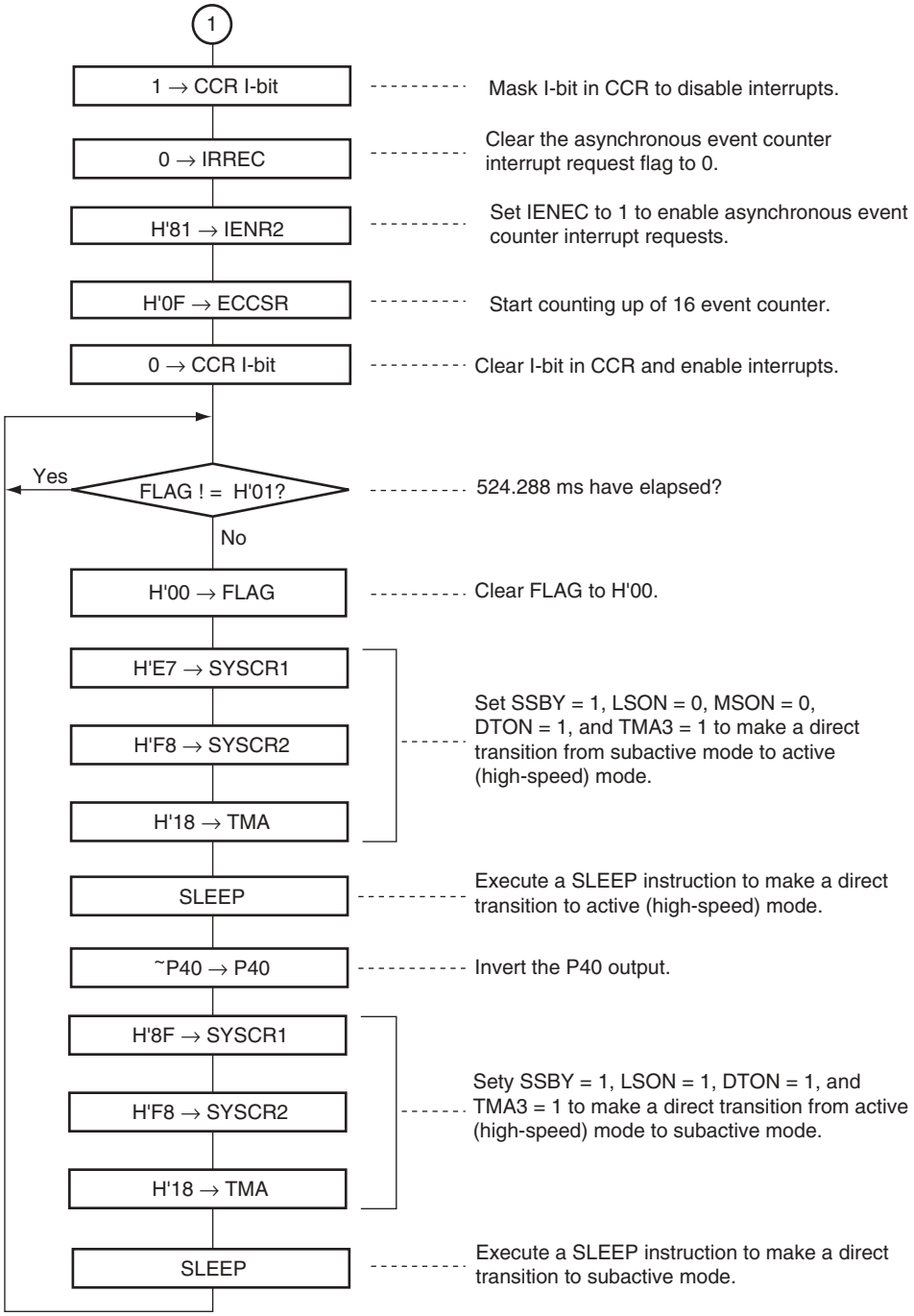
Label	Function	Address	Used in
FLAG	Flag indicating 524.288 ms have elapsed.	H'FB80	main, aecint
CNT	8-bit counter to count the number of timer F interrupt requests.	H'FB81	main, aecint

5. Flowchart

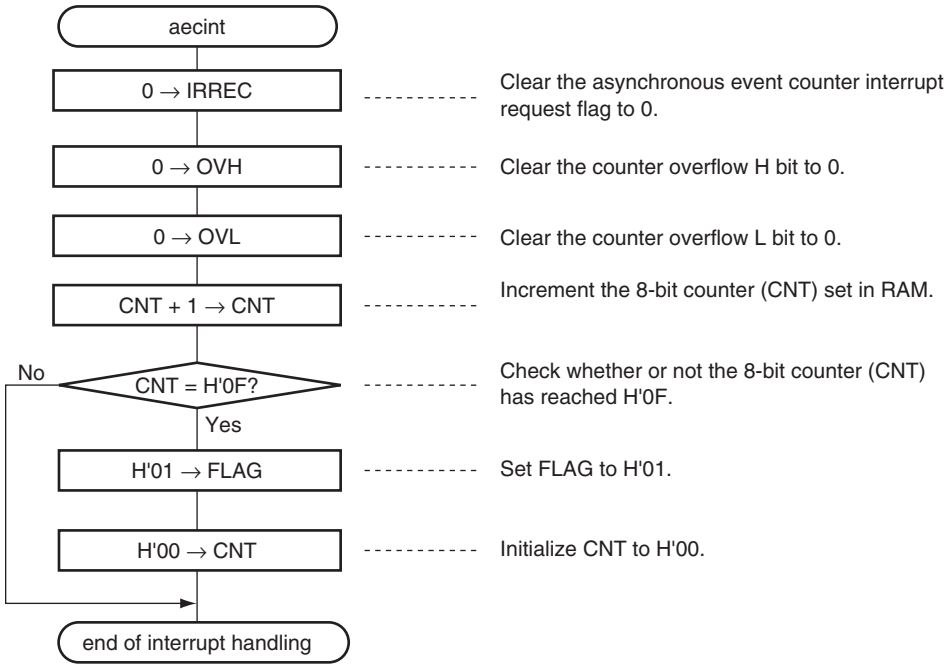
1. Main routine



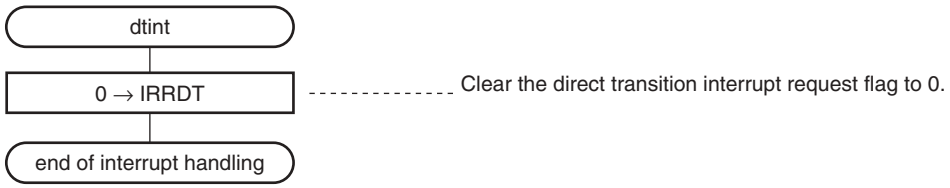
Note: * In this sample task, the stack pointer is set in INIT.SRC (assembly language).


1
1 → CCR I-bit
Mask I-bit in CCR to disable interrupts.
0 → IRREC
Clear the asynchronous event counter interrupt request flag to 0.
H'81 → IENR2
Set IENEC to 1 to enable asynchronous event counter interrupt requests.
H'0F → ECCSR
Start counting up of 16 event counter.
0 → CCR I-bit
Clear I-bit in CCR and enable interrupts.
Yes
FLAG ! = H'01?
524.288 ms have elapsed?
No
H'00 → FLAG
Clear FLAG to H'00.
H'E7 → SYSCR1
H'F8 → SYSCR2
H'18 → TMA
Set SSBY = 1, LSON = 0, MSON = 0, DTON = 1, and TMA3 = 1 to make a direct transition from subactive mode to active (high-speed) mode.
SLEEP
Execute a SLEEP instruction to make a direct transition to active (high-speed) mode.
~P40 → P40
Invert the P40 output.
H'8F → SYSCR1
H'F8 → SYSCR2
H'18 → TMA
Sety SSBY = 1, LSON = 1, DTON = 1, and TMA3 = 1 to make a direct transition from active (high-speed) mode to subactive mode.
SLEEP
Execute a SLEEP instruction to make a direct transition to subactive mode.

2. Asynchronous event count interrupt handling routine.



3. Direct transition interrupt handling routine.



6. Program Listing

INIT.SRC (Program listing)

```
    .EXPORT   _INIT
    .IMPORT   _main
;
    .SECTION  P,CODE
_INIT:
    MOV.W     #H'FF80,R7
    LDC.B     #B'10000000,CCR
    JMP       @_main
;
    .END
```

```
/************************************************************************************************************************/
/*                                                                                                                      */
/*  H8/300L Super Low Power Series                                                                                      */
/*      -H8/38024 Series-                                                                                               */
/*  Application Note                                                                                                    */
/*                                                                                                                      */
/*  'Asynchronous Event Counter Control'                                                                                */
/*                                                                                                                      */
/*  Function                                                                                                            */
/*  : AEC(Asynvhronous Event Counter)                                                                                   */
/*                                                                                                                      */
/*  External Clock : 10MHz                                                                                              */
/*  Internal Clock : 5MHz                                                                                               */
/*  Sub Clock      : 32.768kHz                                                                                          */
/*                                                                                                                      */
/************************************************************************************************************************/

#include    <machine.h>

/************************************************************************************************************************/
/*  Symbol Definition                                                                                                   */
/************************************************************************************************************************/
struct BIT {
    unsigned char   b7:1;       /* bit7 */
    unsigned char   b6:1;       /* bit6 */
    unsigned char   b5:1;       /* bit5 */
    unsigned char   b4:1;       /* bit4 */
    unsigned char   b3:1;       /* bit3 */
    unsigned char   b2:1;       /* bit2 */
    unsigned char   b1:1;       /* bit1 */
    unsigned char   b0:1;       /* bit0 */
};

#define     AEGSR       *(volatile unsigned char *)0xFF92       /* Input Edge Select Register              */
#define     ECCR        *(volatile unsigned char *)0xFF94       /* Event Counter Control Register          */
#define     ECCSR       *(volatile unsigned char *)0xFF95       /*Event Counter Control Status Register    */
#define     ECCSR_BIT   (*(struct BIT *)0xFF95)                 /*Event Counter Control Status Register    */
#define     OVH         ECCSR_BIT.b7                            /* Counter Over Flow H                     */
#define     OVL         ECCSR_BIT.b6                            /* Counter Over Flow L                     */
#define     ECH         *(volatile unsigned char *)0xFF96       /* Event Counter H                         */
#define     ECL         *(volatile unsigned char *)0xFF97       /* Event Counter L                         */
#define     SMR         *(volatile unsigned char *)0xFFA8       /* Serial Mode Register                    */
#define     SMR_BIT     (*(struct BIT *)0xFFA8)                 /* Serial Mode Register                    */
#define     COM         SMR_BIT.b7                              /* Communication Mode                      */
```

```
#define     CHR         SMR_BIT.b6                          /* Character Length                          */
#define     PE          SMR_BIT.b5                          /* Parity Enable                             */
#define     PM          SMR_BIT.b4                          /* Parity Mode                               */
#define     STOP        SMR_BIT.b3                          /* Stop Bit Length                           */
#define     MP          SMR_BIT.b2                          /* Multiprocessor Mode                       */
#define     CKS1        SMR_BIT.b1                          /* Clock Select 1                            */
#define     CKS0        SMR_BIT.b0                          /* Clock Select 0                            */
#define     SCR3        *(volatile unsigned char *)0xFFAA   /* Serial Control Register 3                 */
#define     SCR3_BIT    (*(struct BIT *)0xFFAA)             /* Serial Control Register 3                 */
#define     TIE         SCR3_BIT.b7                         /* Transmit Interrupt Enable                 */
#define     RIE         SCR3_BIT.b6                         /* Receive Interrupt Enable                  */
#define     TE          SCR3_BIT.b5                         /* Transmit Enable                           */
#define     RE          SCR3_BIT.b4                         /* Receive Enable                            */
#define     MPIE        SCR3_BIT.b3                         /* Multiprocessor Interrupt Enable           */
#define     TEIE        SCR3_BIT.b2                         /* Transmit End Interrupt Enable             */
#define     CKE1        SCR3_BIT.b1                         /* Clock Enable 1                            */
#define     CKE0        SCR3_BIT.b0                         /* Clock Enable 0                            */
#define     TMA         *(volatile unsigned char *)0xFFB0   /* Timer Mode Register A                     */
#define     PMR3_BIT    (*(struct BIT *)0xFFCA)             /* Port Mode Register 3                      */
#define     AEVL        PMR3_BIT.b7                         /* P37/AEVL Select                           */
#define     PDR4        *(volatile unsigned char *)0xFFD7   /* Port Data Register 4                      */
#define     PDR4_BIT    (*(struct BIT *)0xFFD7)             /* Port Data Register 4                      */
#define     P40         PDR4_BIT.b0                         /* Port 40                                   */
#define     PCR4        *(volatile unsigned char *)0xFFE7   /* Port Control Register4                    */
#define     PCR4_BIT    (*(struct BIT *)0xFFE7)             /* Port Control Register4                    */
#define     PCR40       PCR4_BIT.b0                         /* Port Control Register40                   */
#define     SYSCR1      *(volatile unsigned char *)0xFFF0   /* System Control Register 1                 */
#define     SYSCR1_BIT  (*(struct BIT *)0xFFF0)             /* System Control Register 1                 */
#define     SSBY        SYSCR1_BIT.b7                       /* Software Standby                          */
#define     STS2        SYSCR1_BIT.b6                       /* Standby Timer Select 2                    */
#define     STS1        SYSCR1_BIT.b5                       /* Standby Timer Select 1                    */
#define     STS0        SYSCR1_BIT.b4                       /* Standby Timer Select 0                    */
#define     LSON        SYSCR1_BIT.b3                       /* Low Speed On Flag                         */
#define     MA1         SYSCR1_BIT.b1                       /* Active Mode Clock Select 1                */
#define     MA0         SYSCR1_BIT.b0                       /* Active Mode Clock Select 0                */
#define     SYSCR2      *(volatile unsigned char *)0xFFF1   /* System Control Register 2                 */
#define     SYSCR2_BIT  (*(struct BIT *)0xFFF1)             /* System Control Register 2                 */
#define     NESEL       SYSCR2_BIT.b4                       /* Noise Elimination Sampling                */
                                                            /*                          Frequency Select */
#define     DTON        SYSCR2_BIT.b3                       /* Direct Transfer On Flag                   */
#define     MSON        SYSCR2_BIT.b2                       /* Middle Speed On Flag                      */
#define     SA1         SYSCR2_BIT.b1                       /* Subactive Mode Clock Select 1             */
#define     SA0         SYSCR2_BIT.b0                       /* Subactive Mode Clock Select 0             */
#define     IENR2       *(volatile unsigned char *)0xFFF4   /* Interrupt Enable Register 2               */
#define     IENR2_BIT   (*(struct BIT *)0xFFF4)             /* Interrupt Enable Register 2               */
#define     IENDT       IENR2_BIT.b7                        /* Timer FH Interrupt Enable                 */
#define     IENEC       IENR2_BIT.b0                        /* Timer FH Interrupt Enable                 */
#define     IRR2_BIT    (*(struct BIT *)0xFFF7)             /* Interrupt Request Register 2              */
#define     IRRDT       IRR2_BIT.b7                         /* Timer FH Interrupt Request Flag           */
#define     IRREC       IRR2_BIT.b0                         /* Timer FH Interrupt Request Flag           */

#pragma interrupt (aecint)
#pragma interrupt (dtint)
```

```
/************************************************************************************************************************/
/*  Function define                                                                                                     */
/************************************************************************************************************************/
extern void INIT ( void );                                      /* SP Set                                               */
void        main ( void );
void        aecint ( void );
void        dtint ( void );

/************************************************************************************************************************/
/*  RAM define                                                                                                          */
/************************************************************************************************************************/
unsigned char   FLAG;                                           /* User Flag Area                                       */
unsigned char   CNT;                                            /* User Flag Area                                       */

/************************************************************************************************************************/
/*  Vector Address                                                                                                      */
/************************************************************************************************************************/
#pragma section     V1                                          /* Vector Section Set                                   */
void (*const VEC_TBL1[])(void) = {
    INIT                                                        /* 0x0000 Reset Vector                                  */
};
#pragma section     V2                                          /* Vector Section Set                                   */
void (*const VEC_TBL2[])(void) = {
    aecint                                                      /* 0x0018 Timer F Interrupt Vector                      */
};
#pragma section     V3                                          /* Vector Section Set                                   */
void (*const VEC_TBL3[])(void) = {
    dtint                                                       /* 0x0028 Timer F Interrupt Vector                      */
};

#pragma section                                                 /* P                                                    */
/************************************************************************************************************************/
/*  Main Program                                                                                                        */
/************************************************************************************************************************/
void    main ( void )
{
    set_imask_ccr(1);                                           /* Interrupt Disable                                    */

    CNT = 0;                                                    /* Initialize 8-bit Counter                             */
    FLAG = 0;                                                   /* Initialize Event Flag                                */

    SCR3 = 0;
    COM = 0;
    PCR40 = 1;                                                  /* Initialize P40 Terminal Function                     */
    PDR4 = 0xF8;                                                /* Initialize P40 PDR                                   */

    AEVL = 1;
    AEGSR = 0x00;
    ECCR = 0x00;
    ECCSR = 0;                                                  /* Reset 16-bit Event Counter                           */

    SYSCR1 = 0x8F;                                              /* Set SYSCR1                                           */
    SYSCR2 = 0xF8;                                              /* Set SYSCR2                                           */
    TMA = 0x18;                                                 /* Initialize TCA Overflow Period                       */

    IRRDT = 0;
    IENR2 = 0x80;                                               /* Timer A Interrupt Enable                             */

    set_imask_ccr(0);                                           /* Interrupt Enable                                     */
```

```
    sleep();                                        /* Transition to Sleep Mode          */

    set_imask_ccr(1);                               /* Interrupt Disable                 */

    IRREC = 0;
    IENR2 = 0x81;                                   /* Timer A Interrupt Enable          */
    ECCSR = 0x0F;

    set_imask_ccr(0);                               /* Interrupt Enable                  */

    while(1){
        while (!FLAG);

        FLAG = 0;
        SYSCR1 = 0xE7;
        SYSCR2 = 0xF8;
        TMA = 0x18;                                 /* TMA3 = "1"                        */

        sleep();                                    /* Transition to Sleep Mode          */
        P40 = ~P40;

        SYSCR1 = 0x8F;                              /* Set SYSCR1                        */
        SYSCR2 = 0xF8;                              /* Set SYSCR2                        */
        TMA = 0x18;                                 /* TMA3 = "1"                        */
        sleep();
    }
}

/*****************************************************************************************************************/
/*  AEC Interrupt                                                                                                */
/*****************************************************************************************************************/
void aecint ( void )
{
    IRREC = 0;                                      /* Clear IRREC                       */

    OVH = 0;                                        /* Clear OVH                         */
    OVL = 0;                                        /* Clear OVL                         */
    CNT++;                                          /* Increment CNT                     */

    if ( CNT > 0x0F ){
        FLAG = 1;                                   /* Set Event Flag                    */
        CNT = 0;                                    /* Initialize 8-bit Counter          */
    }
}
```

```
/*****************************************************************************************************************/
/*  Direct Transfer Interrupt                                                                                    */
/*****************************************************************************************************************/
void    dtint ( void )
{
    IRRDT = 0;                                          /* Clear IRRDT                          */
}
```

Link address specifications

Section Name	Address
CV1	H'0000
CV2	H'0018
CV3	H'0028
P	H'0100
B	H'FB80

Revision Record

		Description	
Rev.	Date	Page	Summary
1.00	Dec.19.03	—	First edition issued

Keep safety first in your circuit designs!

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

Notes regarding these materials

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