

S1C17 Family Port Nested Interrupt Application Notes

Evaluation board/kit and Development tool important notice

1. This evaluation board/kit or development tool is designed for use for engineering evaluation, demonstration, or development purposes only. Do not use it for other purposes. It is not intended to meet the requirements of design for finished products.
2. This evaluation board/kit or development tool is intended for use by an electronics engineer and is not a consumer product. The user should use it properly and in a safe manner. Seiko Epson does not assume any responsibility or liability of any kind of damage and/or fire caused by the use of it. The user should cease to use it when any abnormal issue occurs even during proper and safe use.
3. The part used for this evaluation board/kit or development tool may be changed without any notice.

NOTICE

No part of this material may be reproduced or duplicated in any form or by any means without the written permission of Seiko Epson. Seiko Epson reserves the right to make changes to this material without notice. Seiko Epson does not assume any liability of any kind arising out of any inaccuracies contained in this material or due to its application or use in any product or circuit and, further, there is no representation that this material is applicable to products requiring high level reliability, such as, medical products. Moreover, no license to any intellectual property rights is granted by implication or otherwise, and there is no representation or warranty that anything made in accordance with this material will be free from any patent or copyright infringement of a third party. When exporting the products or technology described in this material, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You are requested not to use, to resell, to export and/or to otherwise dispose of the products (and any technical information furnished, if any) for the development and/or manufacture of weapon of mass destruction or for other military purposes.

All brands or product names mentioned herein are trademarks and/or registered trademarks of their respective companies.

Table of Contents

1. Overview	1
1.1 Functions.....	1
1.2 Folder Configuration.....	2
1.3 File Configuration	2
2. Details of Sample Software Functions	3
2.1 Sample Software Specifications	3
2.2 Processing Overview	4
2.3 Using in Applications.....	5
3. Function Specifications	7
4. Operating Time	8
Revision History.....	9

1. Overview

The S1C17 Family microcontrollers have an interrupt vector table in which multiple interrupt sources for each peripheral circuit are combined into each interrupt vector. For the port input interrupt, the interrupt sources for all ports are assigned to one interrupt vector. Therefore, controlling only the interrupt controller does not allow nested interrupt handling with prioritized port interrupt sources.

This application note provides the functions that manage port input interrupt priorities and handle nested interrupts.

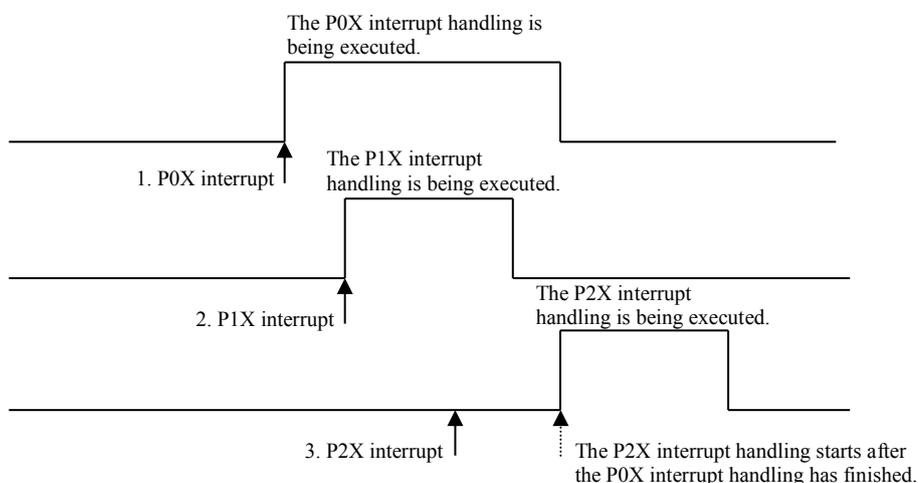
1.1 Functions

A function in this application note sets the interrupt level for each port group (P0X, P1X, P2X, ...).

The interrupt handler function in this application note handles port input interrupts. If a new interrupt having a higher priority occurs while the function is handling an interrupt, the function executes handling of the new interrupt. The interrupt handling being executed currently is temporarily suspended until the handling of new interrupt has finished. If the new interrupt has a lower or the same priority, it will be handled after the handling of the interrupt being executed has finished.

The figure below shows an example of a port input interrupt handling sequence.

Interrupt levels: P0X = 2
P1X = 3
P2X = 1



1. A P0X interrupt occurs.
→ The P0X interrupt handling is executed.
2. A P1X interrupt occurs while the P0X interrupt handling is being executed.
→ Since the interrupt levels have been set as $P0X < P1X$, the P1X interrupt handling is executed.
3. A P2X interrupt occurs while the P1X interrupt handling is being executed.
→ Since the interrupt levels have been set as $P2X < P1X$ and $P2X < P0X$, the P2X interrupt handling is executed after the P0X interrupt handling has finished.

1. Overview

1.2 Folder Configuration

The folder configuration in this package is as follows:

+c17_Port_Nested_Interrupt
 +c17_Port_Nested_Interrupt_gnu17v2 : Sample program for GNU17 Ver. 2.x
 +c17_Port_Nested_Interrupt_gnu17v3 : Sample program for GNU17 Ver. 3.x
 -s1c17_Port_Nested_Interrupt_j.pdf : Manual (Japanese)
 -s1c17_Port_Nested_Interrupt_e.pdf : Manual (English)
 -License_e.txt : Software license agreement (English)
 -README.txt : README file

1.3 File Configuration

The file configuration in this package is as follows:

File name	Description
src/c17_Port_Nested_Interrupt.c	Port nested interrupt handler function
src/c17_Port_Nested_Interrupt.h	Port nested interrupt handler function header definition file
src/main.c	Main function
src/boot.c	Startup module
src/c17_port_m30_m31_m32_m33_m34.c	Sample port driver
src/c17_port_m30_m31_m32_m33_m34.h	Sample port driver header definition file
src/c17_misc.c	Sample MISC driver
src/c17_misc.h	Sample MISC driver header definition file
src/c17_clg.c	Sample CLG driver
src/c17_clg.h	Sample CLG driver header definition file
src/c17_init_config.h	Peripheral circuit configuration header definition file
src/crt0.h	Interrupt vectors and boot function header definition file
inc	CPU register definition file directory
inc/reg	CPU peripheral circuit control register definition file directory

2. Details of Sample Software Functions

This chapter gives a detailed description of the sample software functions.

2.1 Sample Software Specifications

This sample software is implemented for S1C17M33 to execute the following processing:

- Configures the specified ports as input ports that generate an interrupt when the input signal transits from a high level to a low level.
- When an interrupt occurs from an input port, the sample software configures the specified port as an output port that outputs a high level.
- Waits for the input port to be changed to a high level.
- Sets the output port to output a low level.

The table below lists the configuration of the ports used in this sample software.

Table 2.1.1 Port Configuration in Sample Software

Port group	Configuration
P0X	Input interrupt port = P00
	Output port = P07
	Interrupt level = 2
P1X	Input interrupt port = P10
	Output port = P17
	Interrupt level = 3
P2X	Input interrupt port = P20
	Output port = P27
	Interrupt level = 2
P3X	Input interrupt port = P30
	Output port = P37
	Interrupt level = 1

2. Details of Sample Software Functions

2.2 Processing Overview

Main processing

The main function initializes the ports as follows:

- Sets the interrupt level for the P0X port group.
- Enables the P00 port to generate an interrupt at the falling edge of the input signal.
- Sets the P07 port to output a low level.
- Sets the interrupt level for the P1X port group.
- Enables the P10 port to generate an interrupt at the falling edge of the input signal.
- Sets the P17 port to output a low level.
- Sets the interrupt level for the P2X port group.
- Enables the P20 port to generate an interrupt at the falling edge of the input signal.
- Sets the P27 port to output a low level.
- Sets the interrupt level for the P3X port group.
- Enables the P30 port to generate an interrupt at the falling edge of the input signal.
- Sets the P37 port to output a low level.

Interrupt handling (P0X)

The interrupt handler calls the port group interrupt function (`intInputP0x`) that performs the processing shown below.

- Sets the P07 port to output a high level.
- Waits for the P00 port to be changed to a high level.
- Sets the P07 port to output a low level.

The interrupt functions for other port groups (`intInputP1x`, `intInputP2x`, ...) performs the same processing as above except for the port numbers.

2.3 Using in Applications

This section describes how to import the sample software into your application project.

1. Add header file to project
Add “c17_Port_Nested_Interrupt.c/c17_Port_Nested_Interrupt.h” to the project.
2. Declare header file to be included
Add the #include declaration shown below in the source file that refers to the sample functions.

```
#include "c17_port_nested_interrupt.h"
```

3. Register to vector table

When GNU17 Ver. 2.x is used

Register the interrupt function “c17portNestedInterruptPortInput” as the PORT interrupt vector.

boot.c

```
#include "c17_port_nested_interrupt.h"
/** vector table */
func *const vector[] = {
    /* ----- Non-Maskable Interrupts ----- */
    VECTOR(boot), // 0 00 Reset
    VECTOR(intAddrErr), // 1 04 Address misaligned
    VECTOR(intReserved), // 2 08 NMI
    VECTOR(emu_copro_process), // 3 0C Co-processor emulation

    /* ----- Maskable Interrupts ----- */
    VECTOR(c17intSvd), // 4 10 SVD3 interrupt
    VECTOR(c17portNestedInterruptPortInput), // 5 14 PORT interrupt
};
```

When GNU17 Ver. 3.x is used

When using the sample port driver, comment out the vector handler definition shown below.

c17_port_m30_m31_m32_m33_m34.c

```
/* --- interrupt handler (GNU17v3 Only) --- */
#ifdef CRT0_LINKED
//C17_INTERRUPT_HANDLER(_vector05_handler, c17intPortInput); // comment out
#endif /* CRT0_LINKED */
```

2. Details of Sample Software Functions

4. Add user interrupt processing

Add user processing routines to the interrupt functions (intInputP0x, intInputP1x, intInputP2x, ...) written in the “c17_port_nested_interrupt.c” file.

```
void intInputP0x(void)
{
    unsigned short interruptFlag;

    interruptFlag = pxintf_get[C17_PORT_GROUP_P0X]; //Obtain interrupt source storage variable
    pxintf_get[C17_PORT_GROUP_P0X] = 0; //Clear interrupt source storage variable
    asm("ei"); //Enable nested interrupts
    if(interruptFlag & 0x01){
        /**
        * @todo add interrupt function of P00 //Add user processing routine
        */
    }
    if(interruptFlag & 0x02){
        /**
        * @todo add interrupt function of P01 //Add user processing routine
        */
    }
};
```

5. Set interrupt level for each port group

Use the “c17portNestedInterruptSetInterruptLevel” function to set the interrupt level for the port group (P0X, P1X, P2X, ...) of which input interrupt is used.

```
c17portNestedInterruptSetInterruptLevel(C17_PORT_GROUP_P0X, 2);
c17enableIntP0x(0, C17_PORT_INT_EDGE_FALLING);

c17portNestedInterruptSetInterruptLevel(C17_PORT_GROUP_P1X, 3);
c17enableIntP1x(0, C17_PORT_INT_EDGE_FALLING);

c17portNestedInterruptSetInterruptLevel(C17_PORT_GROUP_P2X, 2);
c17enableIntP2x(0, C17_PORT_INT_EDGE_FALLING);

c17portNestedInterruptSetInterruptLevel(C17_PORT_GROUP_P3X, 1);
c17enableIntP3x(0, C17_PORT_INT_EDGE_FALLING);
```

3. Function Specifications

This chapter describes the functions in this sample software.

Port group interrupt level setting function

Function name	c17portNestedInterruptSetInterruptLevel	
Arguments	unsigned short portGr	Port group (P0X = 0, P1X = 1, ...)
	unsigned short portIntLevel	Interrupt level from 1 to 4
Return value	unsigned short	EXIT_SUCCESS: Successful EXIT_FAILURE: Failed
Description	This function sets the interrupt level for the port group specified with the argument.	
Remarks	The effective range of the interrupt level is between 1 to 4.	

Port interrupt handler function

Function name	c17portNestedInterruptPortInput
Argument	None
Return value	None
Description	This function is the port interrupt handler that calls an interrupt function (intInputP0x, intInputP1x, intInputP2x, ...) according to the port group that generates an interrupt.
Remarks	When GNU17 Ver. 2.x is used, register this function as the PORT interrupt vector.

Port group interrupt functions

Function names	intInputP0x, intInputP1x, intInputP2x, ...
Argument	None
Return value	None
Description	These functions are called from the port interrupt handler function and perform the following processing: <ol style="list-style-type: none"> 1. Obtains the interrupt source from the interrupt source storage variable. 2. Clears the interrupt source storage variable. 3. Enables nested port interrupts. 4. Executes the processing for the port that generates an interrupt (user defined routine).
Remarks	The processing routine for each port should be described by the user.

4. Operating Time

4. Operating Time

The time shown below will be taken after a hardware interrupt occurs from an input port until the port interrupt function (intInputPxx) is called.

Time: 228 μ s

Reference

Normal S1C17M33 sample software

Time: 208 μ s

Conditions

Operating clock: IOOSC = 700 kHz

Access wait cycle: FLASHCWAIT.RDWAIT[1:0] bits = 0x0 (no wait)

Input interrupt port: P10

AMERICA

EPSON ELECTRONICS AMERICA, INC.

214 Devcon Drive,
San Jose, CA 95112, USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

EUROPE

EPSON EUROPE ELECTRONICS GmbH

Riesstrasse 15, 80992 Munich,
GERMANY
Phone: +49-89-14005-0 FAX: +49-89-14005-110

ASIA

EPSON (CHINA) CO., LTD.

4th Floor, Tower 1 of China Central Place, 81 Jianguo Road, Chaoyang
District, Beijing 100025 China
Phone: +86-10-8522-1199 FAX: +86-10-8522-1120

SHANGHAI BRANCH

Room 1701 & 1704, 17 Floor, Greenland Center II,
562 Dong An Road, Xu Hui District, Shanghai, CHINA
Phone: +86-21-5330-4888 FAX: +86-21-5423-4677

SHENZHEN BRANCH

Room 804-805, 8 Floor, Tower 2, Ali Center, No.3331
Keyuan South RD (Shenzhen bay), Nanshan District, Shenzhen
518054, CHINA
Phone: +86-10-3299-0588 FAX: +86-10-3299-0560

EPSON TAIWAN TECHNOLOGY & TRADING LTD.

14F, No. 7, Song Ren Road,
Taipei 110, TAIWAN
Phone: +886-2-8786-6688 FAX: +886-2-8786-6660

EPSON SINGAPORE PTE., LTD.

1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

SEIKO EPSON CORP.**KOREA OFFICE**

19F, KLI 63 Bldg., 60 Yoido-dong,
Youngdeungpo-Ku, Seoul 150-763, KOREA
Phone: +82-2-784-6027 FAX: +82-2-767-3677

SEIKO EPSON CORP.**SALES & MARKETING DIVISION****Device Sales & Marketing Department**

421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5816 FAX: +81-42-587-5116