

S1C17801 ADC Application Note

When using the commands, follow the instructions of NOTICE_Application Notes Sample Programs.pdf being included in the downloaded compressed file.

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. This material or portions thereof may contain technology or the subject relating to strategic products under the control of the Foreign Exchange and Foreign Trade Law of Japan and may require an export license from the Ministry of Economy, Trade and Industry or other approval from another government agency.

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

OVERVIEW

This document is a reference to use the ADC function of S1C17801.

OPERATING ENVIRONMENT

- S5U1C17801T1100 (hereafter SVT17801:Software eValuation Tool for S1C17801)
SVT17801 CPU board and SVT17 ICD board
- USB miniB cable
- PC
 - Installed with GNU17 development tools
 - Installed with the USB driver for the SVT17 ICD board

Table of Contents

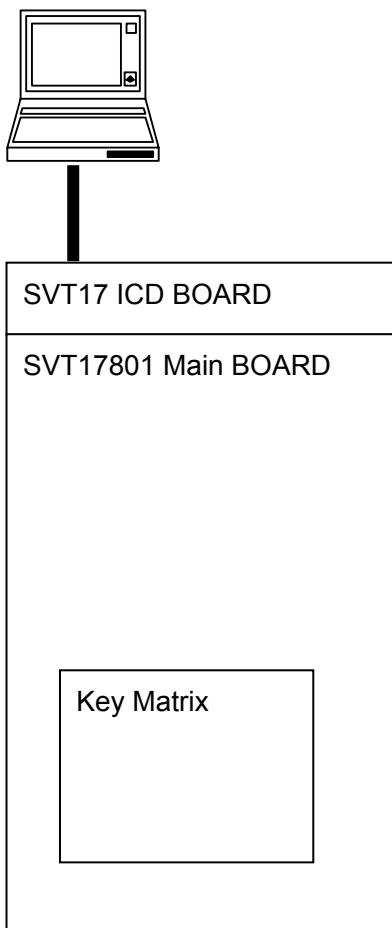
1. SPECIFICATIONS	1
2. DESCRIPTIONS OF FUNCTIONS USED	2
3. OPERATION.....	3
4. SOFTWARE DESCRIPTION	4
4.1 File Configuration	4
4.2 Descriptions of the Modules	5
4.3 Global Variables	5
4.4 Structure	6
4.5 Operating Procedures	7
4.6 Outline of the sample program operations.....	7
4.7 Flowchart.....	8
4.8 Detailed description of ABC driver.....	10
4.9 Header Definitions	24
REVISION HISTORY	27

1. SPECIFICATIONS

This sample converts input on the Push-Switch (S1 to S6) and Push (SW2) of RotarySW connected to AIN7 on the SVT108/501 board, and discriminate the Switch being pressed.

A pressed Switch is discriminated and if corresponding Switch is found, the result is output to the simulated I/O. The figure below shows the connection between the PC and the SVT17801.

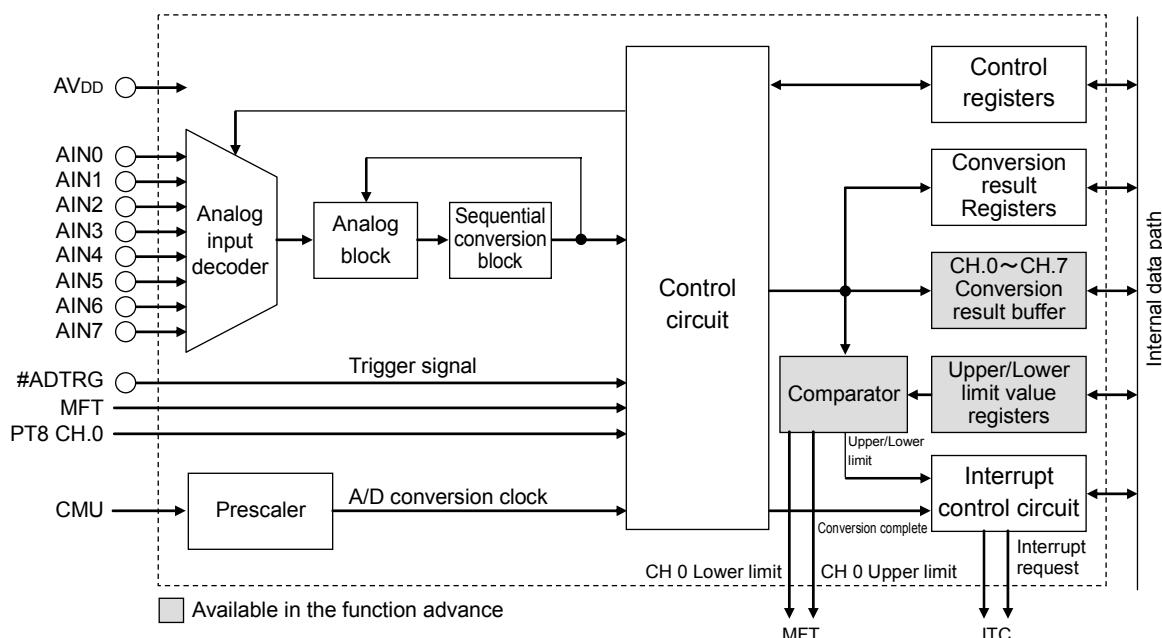
PC (Installed with GNU17 development tools)



2. DESCRIPTIONS OF FUNCTIONS USED

2. DESCRIPTIONS OF FUNCTIONS USED

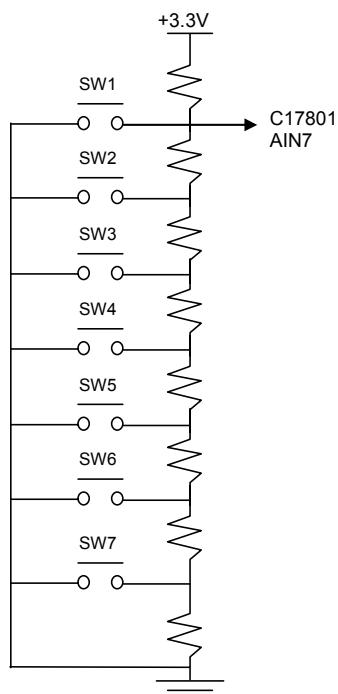
Input and output pins	ADC and analog pins are connected between AIN0 to AIN4 and the connector J7 on the sensor board. AIN5 to AIN6 are connected to the LCD board connector (J13). AIN7 is connected to S1 to S6 and SW2-S. This port is also used by general input and output port pins (P00 to P07).
ADC	Uses AIN7 on ADC.
PT8	Uses channel 10 of PT8. The oscillator generates an 48MHz clock, so the sample program sets PCLK to 1/4096 and the counter reload value to 0xFF.
Interrupt	The vector number and the vector address of the ADC out of range interrupt are as follows: Vector number: 10(0x0a) Vector address: 0x900028 The vector number and the vector address of the ADC conversion complete interrupt are as follows: Vector number: 11(0x0b) Vector address: 0x90002c
	The vector number and the vector address of the PT8 channel 10 interrupt are as follows: Vector number: 21(0x15) Vector address: 0x900054



3. OPERATION

The following describes key connections mounted on the SVT17801 board.

The SW1 to SW6 in the figure below correspond to S1 to S6 on the board, while SW7 to the Push of SW2 on the board.



Pressing one of the above switches changes electric current, which is read as an input value.
This sample program cannot read multiple switch inputs simultaneously.

4. SOFTWARE DESCRIPTION

4. SOFTWARE DESCRIPTION

4.1 File Configuration

File name	Function
boot.c	Startup module
inthdlr.c	Interrupt handler
main.c	Main process of the ADC sample program
vector.c	Vector table
header/reg_801.h	Register definitions
header/vector.h	Vector table definitions
adc_drv/adc_api.h	ADC driver API definitions
adc_drv/adc_drv.c	ADC driver API group
cmu_drv/	CMU driver folder
flashc_drv/	FLASHC driver folder
gpio_drv/	GPIO driver folder
itc_drv/	ITC driver folder
sramc_drv/	SRAMC driver folder
t8_drv/	T8 driver folder
adc_gnu17IDE.lds	Linker script file
adc_gnu17IDE.cmd	GDB command file
adc_gnu17IDE.par	Parameter configuration file
adc_gnu17IDE.mak	Makefile
.cdtproject	Project file
.gnu17project	Project file
.project	Project file
GDB17 Launch for adc.launch	Project startup file

4.2 Descriptions of the Modules

File name: main.c

Function name	Function
main	Configures T8/Port/ITC/ADC and interrupts, and discriminates input switches.
T8Initialize	Starts timer operation after initializing T8-ch0 and setting supplied clk/timer value.
ItcInitialize	Allows the ADC/T8 interrupt flag.
PortInitialize	Port definitions for AD. Uses AIN7
AdcInitialize	Configures channels and other spec used by ADC
InterruptEnableADC	Enable ITC to execute ADC interrupt
InterruptDisableADC	Disable ITC to execute ADC interrupt

For the T8 driver functions, refer to the “Detailed description of T8 Driver” chapter.

For ITC driver functions, refer to the “Detailed description of ITC driver” chapter.

For the GPIO driver functions, refer to the “Detailed description of GPIO driver” chapter.

For the CMU driver functions, refer to the “Detailed description of CMU driver” chapter.

For the FLASHC driver functions, refer to the “Detailed description of FLASHC Driver” chapter.

For SRAMC driver functions, refer to “Detailed descriptions of SRAMC driver” chapter.

4.3 Global Variables

The following shows the global variables used in the sample program.

Variable name	Type	Function
AdcBuffer[1]	unsigned short	Storage buffer for AD conversion

4. SOFTWARE DESCRIPTION

4.4 Structure

Definition name		
T_ADC_CFG config		
Member		
bufferMax	unsigned short	buffer size(half-word:2Byte)
_reserved0	unsigned short	padding
buffer	unsigned short*	Convert result save buffer
intEnableFunc	void(*)(void)	Interrupt Enable function
intDisableFunc	void(*)(void)	Interrupt Disable function
_reserved1	unsigned short	advance mode
advanceMode	unsigned short	advance mode
_reserved2	unsigned short	advance mode
clockDivide	unsigned short	A/D converter clock division ratio select
clockControl	unsigned short	A/D converter clock control
_reserved3	unsigned short	padding
_reserved4	unsigned short	padding
triggerSelect	unsigned short	A/D conversion trigger selection
modeSelect	unsigned short	A/D conversion mode selection
_reserved5	unsigned short	padding
channelStart	unsigned short	A/D converter start channel selection
channelEnd	unsigned short	A/D converter end channel selection
_reserved6	unsigned short	padding
_reserved7	unsigned short	padding
convertIntEnable	unsigned short	Conversion complete interrupt enable (ADV)
compareIntEnable	unsigned short	Out-of-range interrupt enable (ADV)
intSignalMode	unsigned short	Interrupt signal mode (ADV)
_reserved8	unsigned short	padding
samplingTime	unsigned short	Input signal sampling time setup
_reserved9	unsigned short	padding
compareChannle	unsigned short	Upper/Lower limit comparision channel selection (ADV)
compareEnable	unsigned short	Upper/Lower limit comparision enable (ADV)
upperLimit	unsigned short	A/D conversion upper limit value (ADV)
_reserved10	unsigned short	padding
lowerLimit	unsigned short	A/D conversion lower limit value (ADV)
_reserved11	unsigned short	padding
intEnable0	unsigned short	Ch0 conversion complete interrupt enable (ADV)
intEnable1	unsigned short	Ch1 conversion complete interrupt enable (ADV)
intEnable2	unsigned short	Ch2 conversion complete interrupt enable (ADV)
intEnable3	unsigned short	Ch3 conversion complete interrupt enable (ADV)
intEnable4	unsigned short	Ch4 conversion complete interrupt enable (ADV)
intEnable5	unsigned short	Ch5 conversion complete interrupt enable (ADV)
intEnable6	unsigned short	Ch6 conversion complete interrupt enable (ADV)
intEnable7	unsigned short	Ch7 conversion complete interrupt enable (ADV)
_reserved12	unsigned short	padding
Remarks		
Structure for setting the initial value of ADC.		

4.5 Operating Procedures

Import the project

(1) Launch the IDE and import the “abc” project.

※ For information on importing projects, refer to “3. Software Development Procedures” in S5U1C17001C Manual.

Build

(1) Build the “abc” project by using the IDE.

Connection and powering on procedures

(1) Connect SVT17801, USB miniB cable and PC.

(2) Reset SVT17 ICD board.

Run the program

(1) Execute “abc” project by using the IDE.

(2) After “abc” has been executed, a message “This is sample program for AD controller.” is output to the simulated I/O.

(3) Press one of Push Switch S1 to S6 or SW2.

A message that the corresponding Switch has been pressed is output to the Simulated I/O.

4.6 Outline of the sample program operations

(1) Initialize the PT8.

(2) Set the Port0 for analog input.

(3) Initialize ITC. (Enable interrupt for ADC/PT8Ch0.)

(4) Initialize the ADC.

(5) Enable the CPU interrupt.

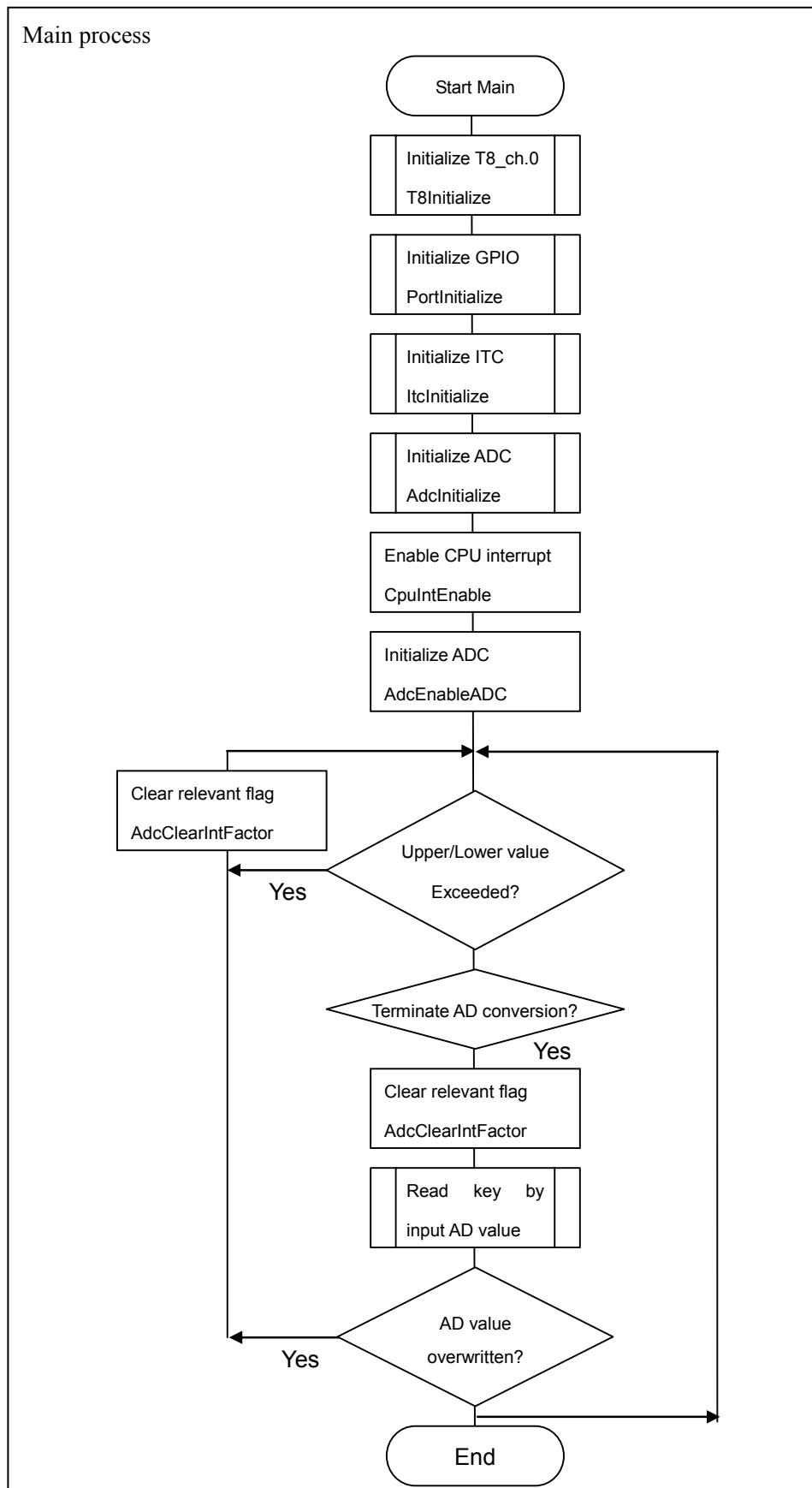
(6) Enable ADC functions.

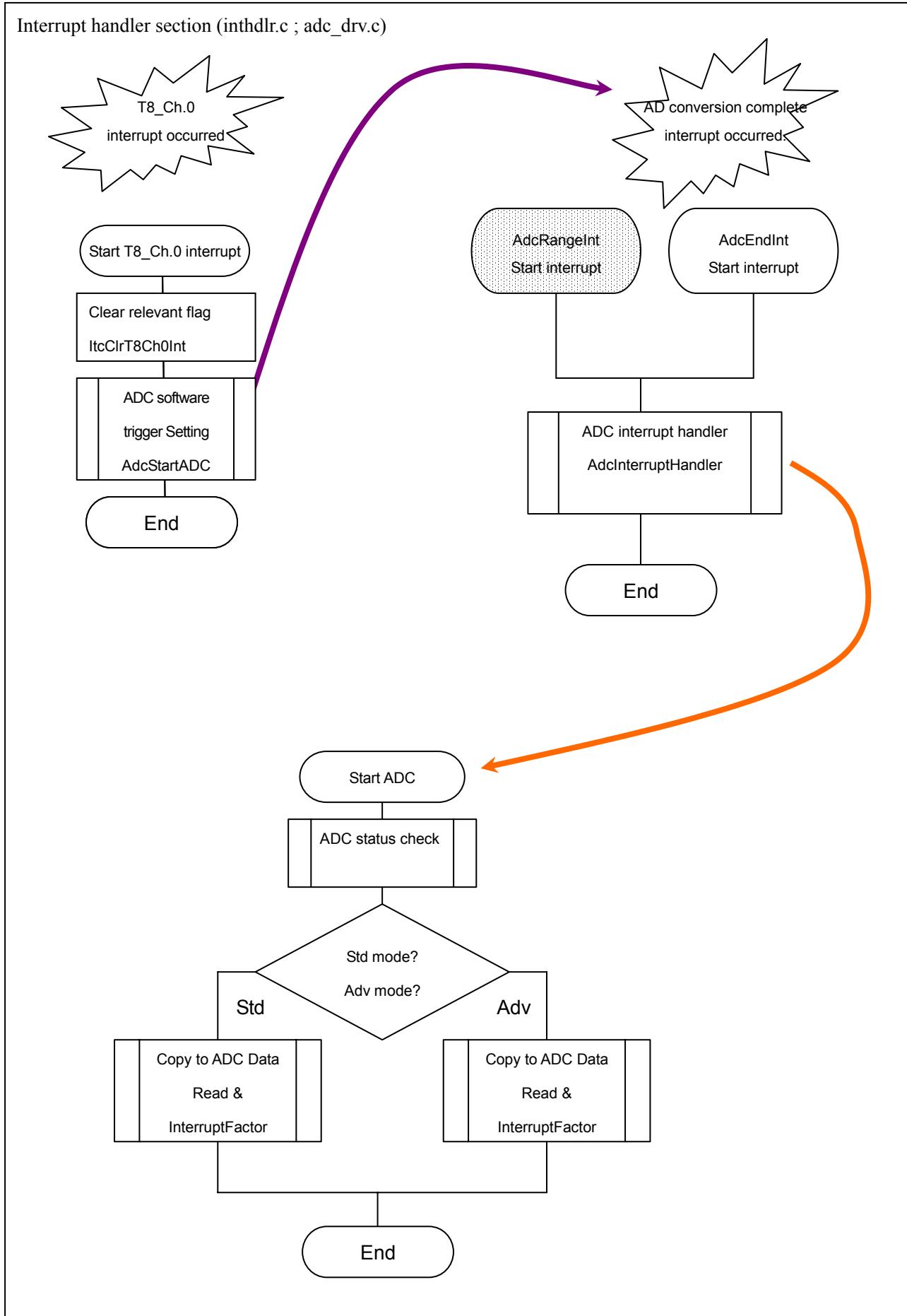
(7) Wait for a conversion result reported from AIN7 and, if the result is within the defined range, output the relevant message to the simulated I/O.

4. SOFTWARE DESCRIPTION

4.7 Flowchart

The following shows the flowchart of the main routine and interrupt handler functions.





4. SOFTWARE DESCRIPTION

4.8 Detailed description of ABC driver

The following provides descriptions on the functions described in the files adc_drv.c and adc_api.h.

Initializing ADC

Format	Int AdcInit(T_ADC_CFG *config)																																	
Function	Initializes the ADC functions																																	
Argument	config -in Parameters setting AD controller																																	
Return value	SUCCESS : Terminated successfully ERROR : Terminated abnormally																																	
(Description)																																		
Initializes ADC according to the config details. The following shows the structure of T_ADC_CFG. When using this function for configuration, define the setting value for each member. Undefined value is set for a member that has not been defined. (ADV) can be defined when AdvMode is enabled.																																		
<pre>typedef struct { unsigned short bufferMax; // The maximum number of buffer (by half-word unit) unsigned short *buffer; // Buffer holding the covers result // [15]:Overwrite Error // [11]:Conversion result valid flag // [9:0]:Conversion result void (*intEnableFunc)(void); // Interrupt Enable function void (*intDisableFunc)(void); // Interrupt Disable function unsigned short advanceMode:1; // advance mode unsigned short clockDivide:3; // A/D converter clock division ratio select unsigned short clockControl:1; // A/D converter clock control unsigned short triggerSelect:2; // A/D conversion trigger selection unsigned short modeSelect:1; // A/D conversion mode selection unsigned short channelStart:3; // A/D converter start channel selection unsigned short channelEnd:3; // A/D converter end channel selection unsigned short convertIntEnable:1; // Conversion complete interrupt enable(ADV) unsigned short compareIntEnable:1; // Out-of-range interrupt enable(ADV) intSignalMode:1; // Interrupt signal mode(ADV) samplingTime:2; // Input signal sampling time setup compareChanne:3; // Upper/Lower limit comparision channel selection(ADV) compareEnable:1; // Upper/Lower limit comparision ebnable(ADV) unsigned short upperLimit:10; // A/D conversion upper limit value(ADV) unsigned short lowerLimit:10; // A/D conversion lower limit value(ADV) unsigned short intEnable0:1; // Ch0 conversion complete interrupt enable(ADV) unsigned short intEnable1:1; // Ch1 conversion complete interrupt enable (ADV) unsigned short intEnable2:1; // Ch2 conversion complete interrupt enable (ADV) unsigned short intEnable3:1; // Ch3 conversion complete interrupt enable (ADV) unsigned short intEnable4:1; // Ch4 conversion complete interrupt enable (ADV) unsigned short intEnable5:1; // Ch5 conversion complete interrupt enable(ADV) unsigned short intEnable6:1; // Ch6 conversion complete interrupt enable(ADV) unsigned short intEnable7:1; // Ch7 conversion complete interrupt enable(ADV) } T_ADC_CFG;</pre>																																		
Data within the buffer are assigned as follows:																																		
<table border="1"><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>OWE</td><td>—</td><td>—</td><td>Valid</td><td>—</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Conversion result</td></tr></table>			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OWE	—	—	Valid	—											Conversion result
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																			
OWE	—	—	Valid	—											Conversion result																			
When bit 11 is defined as Valid, the conversion result is valid. ×To be cleared using application on the upper level. Otherwise, the next result will not be held. When bit 15 is defined as OWE, the conversion result is also valid. However, the converted result has been lost in advance.																																		

Setting ADC operating mode

Format	int AdcSetMode(unsigned short mode)		
Function	Switches the operation mode of ADC function (between standard and advance modes)		
Argument	mode	-in	modes switched ADC_STANDARD : Standard mode ADC_ADVANCE : Advance mode
Return value	SUCCESS : Terminated successfully ERROR : Undefined		
(Description)	<p>The operating mode is set by the AdcInit function at initialization. The operating mode can also be switched at any time besides the initialization.</p> <p>When this function is used to change a setting, disable the AD converter. If this function is called while the AD converter is enabled, the setting is not effected and ERROR is returned.</p>		

Obtaining ADC operating mode (Macro function)

Format	unsigned short AdcGetMode(void)		
Function	Returns the current operating mode of ADC function		
Argument	None		
Return value	ADC_STANDARD : Standard mode ADC_ADVANCE : Advance mode		
(Description)	Returns the operating mode currently set.		

Selecting ADC supply clock

Format	int AdcSetClock(unsigned short clock)		
Function	Selects the ADC supply clock		
Argument	clock	-in	Clock selected ADC_CLOCK_256 : PCLK/256 ADC_CLOCK_128 : PCLK/128 ADC_CLOCK_64 : PCLK/64 ADC_CLOCK_32 : PCLK/32 ADC_CLOCK_16 : PCLK/16 ADC_CLOCK_8 : PCLK/8 ADC_CLOCK_4 : PCLK/4 ADC_CLOCK_2 : PCLK/2
Return value	SUCCESS : Terminated successfully ERROR : Undefined		
(Description)	<p>The operating mode is set by the AdcInit function at initialization. The operating mode can also be switched at any time besides the initialization. Select the clock to supply from Prescaler.</p> <p>When this function is used to change a setting, disable the AD converter and turn off the Clock Control. Otherwise, the setting is not effected and ERROR is returned. After the setting, the Clock Control must be started using AdcClockStart.</p>		

4. SOFTWARE DESCRIPTION

Obtaining ADC supply clock (Macro function)

Format	unsigned short AdcGetClock(void)
Function	Obtains current set value of the supply clock
Argument	None
Return value	ADC_CLOCK_256 : PCLK/256 ADC_CLOCK_128 : PCLK/128 ADC_CLOCK_64 : PCLK/64 ADC_CLOCK_32 : PCLK/32 ADC_CLOCK_16 : PCLK/16 ADC_CLOCK_8 : PCLK/8 ADC_CLOCK_4 : PCLK/4 ADC_CLOCK_2 : PCLK/2
(Description) Obtains the set value currently defined.	

Starting ADC clock supply

Format	int AdcStartClock(void)
Function	Starts the supply clock to ADC function
Argument	None
Return value	SUCCESS : Terminated successfully ERROR : Undefined
(Description) Starts clock supply	
When this function is used to change a setting, disable the AD converter. If this function is called while the AD convert is enabled, the setting is not effected and ERROR is returned.	

Stopping ADC clock supply

Format	int AdcStopClock(void)
Function	Stops the supply clock to ADC function
Argument	None
Return value	SUCCESS : Terminated successfully ERROR : Undefined
(Description) Stops clock supply	
When this function is used to change a setting, disable the AD converter. If this function is called while the AD converter is enabled, the setting is not effected and ERROR is returned.	

Setting ADC conversion channel

Format	int AdcSetChannel(unsigned char start, unsigned char end)		
Function	Sets ADC conversion channel		
Argument	start end	-in -in	Start channel number End channel number ADC_CH0 : Channel 0 ADC_CH1 : Channel 1 ADC_CH2 : Channel 2 ADC_CH3 : Channel 3 ADC_CH4 : Channel 4 ADC_CH5 : Channel 5 ADC_CH6 : Channel 6 ADC_CH7 : Channel 7
Return value	SUCCESS ERROR	: Terminated successfully : Undefined	
(Description) Sets the channel for conversion. The channel is set by the AdcInit function at initialization, and can also be set using this function at any time.			
When this function is used to change a setting, disable the AD converter. If this function is called while the AD converter is enabled, the setting is not effected and ERROR is returned.			

Obtaining the set value of ADC conversion channel

Format	void AdcGetChannel(unsigned char *start, unsigned char *end)		
Function	Obtains the set value of ADC conversion channel		
Argument	*start *end	-out -out	Start channel number End channel number ADC_CH0 : Channel 0 ADC_CH1 : Channel 1 ADC_CH2 : Channel 2 ADC_CH3 : Channel 3 ADC_CH4 : Channel 4 ADC_CH5 : Channel 5 ADC_CH6 : Channel 6 ADC_CH7 : Channel 7
Return value	None		
(Description) Obtains the set value of ADC conversion channel.			

4. SOFTWARE DESCRIPTION

Setting conversion operating mode for ADC

Format	int AdcSetConvertMode(unsigned short mode)	
Function	Sets conversion operating mode for ADC	
Argument	mode	-in Conversion operating mode ADC_CONTINUE: Continue mode ADC_NORMAL : Normal mode
Return value	SUCCESS ERROR	: Terminated successfully : Undefined
(Description) Sets the operation mode for conversion. The mode is set by the AdcInit function at initialization, and can also be set using this function at any time.		
When this function is used to change a setting, disable the AD converter. If this function is called while the AD converter is enabled, the setting is not effected and ERROR is returned.		

Obtaining set value of conversion operating mode for ADC (Macro function)

Format	unsigned short AdcGetConvertMode(void)	
Function	Obtains the set value of conversion operating mode for ADC	
Argument	None	
Return value	ADC_CONTINUE : Continue mode ADC_NORMAL : Normal mode	
(Description) Obtains the conversion operation mode currently set.		

Setting trigger to start ADC conversion

Format	int AdcSetTrigger(unsigned short trigger)	
Function	Sets trigger to start ADC conversion	
Argument	trigger	-in Trigger to start conversion ADC_EXTERN : External trigger (#ADTRG) ADC_TIMER_8 : 8 Bit Timer ADC_TIMER_16 : 16 Bit Timer ADC_SOFTWARE : Software
Return value	SUCCESS ERROR	: Terminated successfully : Undefined
(Description) Sets the trigger to start conversion process. The trigger is set by the AdcInit function at initialization, and can also be set using this function at any time.		
When this function is used to change a setting, disable the AD converter. If this function is called while the AD converter is enabled, the setting is not effected and ERROR is returned.		

Obtaining set value of starting ADC conversion (Macro function)

Format	unsigned short AdcGetTrigger(void)
Function	Obtains the set value of the trigger starting ADC conversion
Argument	None
Return value	ADC_EXTERN : External trigger (#ADTRG) ADC_TIMER_8 : 8 Bit Timer ADC_TIMER_16 : 16 Bit Timer ADC_SOFTWARE : Software
(Description)	Returns the trigger value to start conversion.

Setting ADC interrupt signal mode

Format	int AdcSetInterruptMode(unsigned short mode)
Function	Sets ADC interrupt signal mode
Argument	mode -in Interrupt signal mode ADC_INT_CMPONLY : Conversion complete interrupt only ADC_INT_OR : Conversion complete OR comparison result
Return value	SUCCESS : Terminated successfully ERROR : Undefined
(Description)	Sets the interrupt signal mode. The mode is set by the AdcInit function at initialization, and can also be set using this function at any time. However, ERROR is returned unless Advance mode is enabled. When this function is used to change a setting, disable the AD converter. If this function is called while the AD converter is enabled, the setting is not effected and ERROR is returned.

Obtaining set value of ADC interrupt mode (Macro function)

Format	unsigned short AdcGetInterruptMode(void)
Function	Sets ADC interrupt signal mode
Argument	None
Return value	ADC_INT_CMPONLY : Conversion complete interrupt only ADC_INT_OR : Conversion complete OR comparison result
(Description)	Returns the interrupt signal mode.

Enabling ADC conversion complete interrupt

Format	int AdcEnableConvertInterrupt(void)
Function	Enables ADC conversion complete interrupt
Argument	None
Return value	SUCCESS : Terminated successfully ERROR : Undefined
(Description)	Enables interrupt handler when conversion is completed. The mode is set by the AdcInit function at initialization, and can also be set using this function at any time. However, ERROR is returned unless Advance mode is enabled.

4. SOFTWARE DESCRIPTION

Disabling ADC conversion complete interrupt (Macro function)

Format	void AdcDisableConvertInterrupt(void)
Function	Disables ADC conversion complete interrupt
Argument	None
Return value	None
(Description) Disables interrupt handler when conversion is completed. The mode is set by the AdcInit function at initialization, and can also be set using this function at any time.	

Enabling ADC out of range interrupt

Format	int AdcEnableOutOfRangeInterrupt(void)
Function	Enables ADC out of range interrupt.
Argument	None
Return value	SUCCESS : Terminated successfully ERROR : Undefined
(Description) Enables out of range interrupt handler. The mode is set by the AdcInit function at initialization, and can also be set using this function at any time. However, ERROR is returned unless Advance mode is enabled.	

Disabling ADC out of range interrupt (Macro function)

Format	void AdcDisableOutOfRangeInterrupt(void)
Function	Disables ADC out of range interrupt
Argument	None
Return value	None
(Description) Disables out of range interrupt handler. The mode is set by the AdcInit function at initialization, and can also be set using this function at any time.	

Enabling ADC operation

Format	int AdcEnableADC(void)
Function	Enables the ADC operation
Argument	None
Return value	SUCCESS : Terminated successfully ERROR : Undefined
(Description) The ADC operation is enabled, and is started by the trigger input previously set. ERROR is returned unless Clock is started.	

Disabling ADC operation (Macro function)

Format	void AdcDisableADC(void)
Function	Disables the ADC operation
Argument	None
Return value	None
(Description)	Disables the ADC operation.

Starting ADC conversion

Format	int AdcStartADC(void)
Function	Starts ADC conversion process.
Argument	None
Return value	SUCCESS : Terminated successfully ERROR : Undefined
(Description)	Starts the ADC conversion process. (Enabled when trigger is set in software.) ERROR is returned unless ADC is enabled.

Stopping ADC conversion (Macro function)

Format	void AdcStopADC(void)
Function	Stops ADC conversion process.
Argument	None
Return value	None
(Description)	Stops the ADC conversion process. (Can forcibly stop regardless of the trigger setting.)

Setting time for sampling ADC input signal

Format	int AdcSetSampling(unsigned short sampling)	
Function	Sets time for sampling ADC input signal	
Argument	sampling -in Time for sampling conversion input signal ADC_SAMPLING_9 : 9Clock ADC_SAMPLING_7 : 7Clock ADC_SAMPLING_5 : 5Clock ADC_SAMPLING_3 : 3Clock	
Return value	SUCCESS : Terminated successfully ERROR : Undefined	
(Description)	Sets time for sampling conversion input signal. The time is set by the AdcInit function at initialization, and can also be set using this function at any time.	
When this function is used to change a setting, disable the AD converter. If this function is called while the AD converter is enabled, the setting is not effected and ERROR is returned.		

4. SOFTWARE DESCRIPTION

Obtaining set value of time for sampling ADC input signal (Macro function)

Format	unsigned short AdcGetSampling(void)
Function	Obtains set value of time for sampling ADC input signal(Macro function)
Argument	None
Return value	ADC_SAMPLING_9 : 9Clock ADC_SAMPLING_7 : 7Clock ADC_SAMPLING_5 : 5Clock ADC_SAMPLING_3 : 3Clock
(Description)	Returns set value of time for sampling conversion input signal.

Sets ADC upper/lower limit comparison channel

Format	int AdcSetCmpCh(unsigned char ch)
Function	Sets ADC upper/lower limit comparison channel
Argument	ch -in Comparison channel ADC_CH0 : Channel 0 ADC_CH1 : Channel 1 ADC_CH2 : Channel 2 ADC_CH3 : Channel 3 ADC_CH4 : Channel 4 ADC_CH5 : Channel 5 ADC_CH6 : Channel 6 ADC_CH7 : Channel 7
Return value	SUCCESS : Terminated successfully ERROR : Undefined
(Description)	Defines channel number used for comparing values set in the upper/lower limit comparison register. The channel number is set by the AdcInit function at initialization, and can also be set using this function at any time. However, ERROR is returned unless Advance mode is enabled. When this function is used to change a setting, disable the AD converter. If this function is called while the AD converter is enabled, the setting is not effected and ERROR is returned.

Obtaining set value of ADC upper/lower channel comparison channel (Macro function)

Format	unsigned char AdcGetCmpCh(void)
Function	Obtains the set value of the ADC upper/lower limit comparison channel
Argument	None
Return value	ADC_CH0 : Channel 0 ADC_CH1 : Channel 1 ADC_CH2 : Channel 2 ADC_CH3 : Channel 3 ADC_CH4 : Channel 4 ADC_CH5 : Channel 5 ADC_CH6 : Channel 6 ADC_CH7 : Channel 7
(Description)	Returns channel number used for comparing values set in the upper/lower limit comparison register.

Setting ADC upper/lower limit comparison value

Format	int AdcSetLimit(unsigned short upper, unsigned short lower)					
Function	Sets ADC upper/lower limit comparison value.					
Argument	upper	-in	upper limit value 0 to 0x3FF			
	lower	-in	lower limit value 0 to 0x3FF			
Return value	SUCCESS : Terminated successfully ERROR : Undefined					
(Description) Sets upper/lower limit comparison value. The value is set by the AdcInit function at initialization, However, ERROR is returned unless Advance mode is enabled.						
When this function is used to change a setting, disable the AD converter. If this function is called while the AD converter is enabled, the setting is not effected and ERROR is returned.						

Obtaining set value of ADC upper/lower limit comparison

Format	void AdcGetLimit(unsigned short *upper, unsigned short *lower)		
Function	Obtains set value of ADC upper/lower limit comparison		
Argument	*upper	-out	upper limit value 0 to 0x3FF
	*lower	-out	lower limit value 0 to 0x3FF
Return value	None		
(Description) Returns set value of upper/lower limit comparison.			

Enabling interrupt

Format	int AdcEnableInterrupt(unsigned short ch)		
Function	Enables ADC interrupt		
Argument	ch	-in	Channel to be enabled (OR setting allowed) ADC_INT_CH0 : Channel 0 ADC_INT_CH1 : Channel 1 ADC_INT_CH2 : Channel 2 ADC_INT_CH3 : Channel 3 ADC_INT_CH4 : Channel 4 ADC_INT_CH5 : Channel 5 ADC_INT_CH6 : Channel 6 ADC_INT_CH7 : Channel 7
Return value	SUCCESS : Terminated successfully ERROR : Undefined		
(Description) Enables interrupt handler. However, ERROR is returned unless Advance mode is enabled.			

4. SOFTWARE DESCRIPTION

Disabling ADC interrupt (Macro function)

Format	void AdcDisableInterrupt(unsigned short ch)	
Function	Disables ADC interrupt	
Argument	ch -in	Channel to be disabled (OR setting allowed) ADC_INT_CH0 : Channel 0 ADC_INT_CH1 : Channel 1 ADC_INT_CH2 : Channel 2 ADC_INT_CH3 : Channel 3 ADC_INT_CH4 : Channel 4 ADC_INT_CH5 : Channel 5 ADC_INT_CH6 : Channel 6 ADC_INT_CH7 : Channel 7
Return value	SUCCESS ERROR	: Terminated successfully : Undefined
(Description)	Disables interrupt handler.	

Obtaining ADC conversion result (Macro function)

Format	unsigned short AdcGetData(void)
Function	Obtains ADC conversion result.
Argument	None
Return value	Returns the ADC conversion result.
(Description)	Returns the ADC conversion result.

Obtaining ADC conversion channel (Macro function)

Format	unsigned char AdcGetChStatus(void)
Function	Obtains ADC conversion channel
Argument	None
Return value	Returns the ADC conversion result. ADC_CH0 : Channel 0 ADC_CH1 : Channel 1 ADC_CH2 : Channel 2 ADC_CH3 : Channel 3 ADC_CH4 : Channel 4 ADC_CH5 : Channel 5 ADC_CH6 : Channel 6 ADC_CH7 : Channel 7
(Description)	Returns the ADC channel currently in conversion process.

Obtaining ADC status (Macro function)

Format	unsigned short AdcGetStatus(void)
Function	Obtains ADC status
Argument	None
Return value	Returns ADC status (OR condition) ADC_STATUS_UPPER : Out of upper limit ADC_STATUS_LOWER : Out of lower limit ADC_STATUS_CMP : Conversion complete ADC_STATUS_RUN : Currently in conversion process ADC_STATUS_ERR : ADD register overwrite error
(Description)	Returns the ADC status.

Clearing ADC overwrite error (Macro function)

Format	void AdcClearOverwriteError(void)
Function	Clears ADC overwrite
Argument	None
Return value	None
(Description)	Clears ADC overwrite error.

Obtaining ADC advance status (Macro function)

Format	unsigned short AdcGetADVStatus (void)
Function	Obtains ADC upper/lower limit status
Argument	None
Return value	Returns ADC advance status (OR condition) ADC_STATUS_CHO_CMP : Ch.0 Conversion complete ADC_STATUS_CH1_CMP : Ch.1 Conversion complete ADC_STATUS_CH2_CMP : Ch.2 Conversion complete ADC_STATUS_CH3_CMP : Ch.3 Conversion complete ADC_STATUS_CH4_CMP : Ch.4 Conversion complete ADC_STATUS_CH5_CMP : Ch.5 Conversion complete ADC_STATUS_CH6_CMP : Ch.6 Conversion complete ADC_STATUS_CHO_CMP : Ch.7 Conversion complete ADC_STATUS_CH0_ERR : Ch.0 overwrite error ADC_STATUS_CH1_ERR : Ch.1 overwrite error ADC_STATUS_CH2_ERR : Ch.2 overwrite error ADC_STATUS_CH3_ERR : Ch.3 overwrite error ADC_STATUS_CH4_ERR : Ch.4 overwrite error ADC_STATUS_CH5_ERR : Ch.5 overwrite error ADC_STATUS_CH0_ERR : Ch.6 overwrite error ADC_STATUS_CH7_ERR : Ch.7 overwrite error
(Description)	Returns the upper/lower limit value obtained from ADC conversion result.

4. SOFTWARE DESCRIPTION

Clearing ADC advance overwrite error (Macro function)

Format	void AdcClearADVOverwriteError(unsigned short over)	
Function	Clears ADC advance overwrite	
Argument	over	-in Overwrite error Ch (OR setting allowed) ADC_STATUS_CH0_ERR : Ch0 Overwrite Error ADC_STATUS_CH1_ERR : Ch1 Overwrite Error ADC_STATUS_CH2_ERR : Ch2 Overwrite Error ADC_STATUS_CH3_ERR : Ch3 Overwrite Error ADC_STATUS_CH4_ERR : Ch4 Overwrite Error ADC_STATUS_CH5_ERR : Ch5 Overwrite Error ADC_STATUS_CH6_ERR : Ch6 Overwrite Error ADC_STATUS_CH7_ERR : Ch7 Overwrite Error
Return value	None	
(Description)	Clears ADC advance overwrite error.	

Obtaining ADC advance conversion result

Format	unsigned short AdcGetADVData(unsigned char ch)	
Function	Obtains ADC advance conversion result	
Argument	ch	-in Channel number ADC_CH0 : Channel 0 ADC_CH1 : Channel 1 ADC_CH2 : Channel 2 ADC_CH3 : Channel 3 ADC_CH4 : Channel 4 ADC_CH5 : Channel 5 ADC_CH6 : Channel 6 ADC_CH7 : Channel 7
Return value	Returns the ADC advance conversion result.	
(Description)	Returns the ADC conversion result.	

Obtaining ADC internal state

Format	void AdcGetInternalState(unsigned char *state, unsigned char *counter)	
Function	Obtaining ADC internal state	
Argument	state	-in Internal state ADC_INTERNAL_ST_IDLE : Idle state ADC_INTERNAL_ST_SAMP : Sampling ADC_INTERNAL_ST_CONV : Currently in conversion process
	counter	-in Internal counter value 0 to 15
Return value	None	
(Description)	Returns the ADC internal state.	

Disables interrupt handler

Format	void AdcInterruptHandler(void)
Function	ADC interrupt handler
Argument	None
Return value	None
(Description) Executes ADC interrupt handler. Holds the conversion result in config.buffer.	

Obtaining ADC internal interrupt flag

Format	unsigned short AdcGetIntFactor(void)
Function	Obtains ADC internal interrupt flag
Argument	None
Return value	ADC_INTR_UPPER_OUT : Out of upper range ADC_INTR_LOWER_OUT : Out of lower range ADC_INTR_CONVERT_CMP : Conversion complete ADC_INTR_OWE : Overwrite
(Description) Returns ADC internal interrupt flag. Stored in the buffer area specified by config.buffer, as Ring Buffer of config.bufferMax. Application on the upper level must call this API to determine whether the conversion process is completed or not. ADC_INTR_CONVERT_CMP/ADC_INTR_OWE is set for each conversion of each channel if conversion for multiple channels is defined. When checking conversion data using this flag, refer to Valid Bit inside config.buffer. If the data is not needed, clear Valid.Bit inside the buffer.	

Clearing ADC internal interrupt flag

Format	void AdcClearIntFactor(unsigned short factor)
Function	Obtains ADC internal interrupt flag
Argument	factor -in Clear flag ADC_INTR_UPPER_OUT : Out of upper range ADC_INTR_LOWER_OUT : Out of lower range ADC_INTR_CONVERT_CMP : Conversion ADC_INTR_OWE : Overwrite
Return value	None
(Description) Clears ADC internal interrupt flag.	

4. SOFTWARE DESCRIPTION

4.9 Header Definitions

The tables below show the definitions used in the driver functions.

The following lists definitions used by config in AdcInit function.

Definition name	Value	Description
ADC_INIT_CONFIG_0	8	Register offset
ADC_INIT_CONFIG_1	(ADC_INIT_CONFIG_0 + 1)	Register offset
ADC_INIT_CONFIG_2	(ADC_INIT_CONFIG_1 + 1)	Register offset
ADC_INIT_CONFIG_3	(ADC_INIT_CONFIG_2 + 1)	Register offset
ADC_INIT_CONFIG_4	(ADC_INIT_CONFIG_3 + 1)	Register offset
ADC_INIT_CONFIG_5	(ADC_INIT_CONFIG_4 + 1)	Register offset
ADC_INIT_CONFIG_6	(ADC_INIT_CONFIG_5 + 1)	Register offset
ADC_INIT_STANDARD	0x00	Standard Mode
ADC_INIT_ADVANCE	0x01	Advance Mode
ADC_INIT_CLK2	0x00	clock Divide 1/2
ADC_INIT_CLK4	0x01	clock Divide 1/4
ADC_INIT_CLK8	0x02	clock Divide 1/8
ADC_INIT_CLK16	0x03	clock Divide 1/16
ADC_INIT_CLK32	0x04	clock Divide 1/32
ADC_INIT_CLK64	0x05	clock Divide 1/64
ADC_INIT_CLK128	0x06	clock Divide 1/128
ADC_INIT_CLK256	0x07	clock Divide 1/256
ADC_INIT_CLKOFF	0x00	clock Control OFF
ADC_INIT_CLKON	0x01	clock Control ON
ADC_INIT_TRIG_SOFT	0x00	trigger Select Software
ADC_INIT_TRIG_16TIMER	0x01	trigger Select 16bit Timer
ADC_INIT_TRIG_8TIMER	0x02	trigger Select 8bit Timer
ADC_INIT_TRIG_ADRG	0x03	trigger Select AD Trigger
ADC_INIT_MODE_NORMAL	0x00	mode Select Normal
ADC_INIT_MODE_CONTINUE	0x01	mode Select Continue
ADC_INIT_CH0	0x00	Channel 0
ADC_INIT_CH1	0x01	Channel 1
ADC_INIT_CH2	0x02	Channel 2
ADC_INIT_CH3	0x03	Channel 3
ADC_INIT_CH4	0x04	Channel 4
ADC_INIT_CH5	0x05	Channel 5
ADC_INIT_CH6	0x06	Channel 6
ADC_INIT_CH7	0x07	Channel 7
ADC_INIT_DISABLE	0x00	Interrupt disable
ADC_INIT_ENABLE	0x01	Interrupt enable
ADC_INIT_INTR_OR	0x00	intSignalMode OR mode
ADC_INIT_INTR_CMP	0x01	intSignalMode Complete only
ADC_INIT_SAMPLING3	0x00	sampling Time Clock3
ADC_INIT_SAMPLING5	0x01	sampling Time Clock5
ADC_INIT_SAMPLING7	0x02	sampling Time Clock7
ADC_INIT_SAMPLING9	0x03	sampling Time Clock9

The following lists definitions used in API functions.

Definition name	Value	Description
ADC_STANDARD	0	Standard Mode
ADC_ADVANCE	ADC_ADVMODE_ADCADV	Advance Mode
ADC_CLOCK_2	ADC_INIT_CLK2	clock Divide 1/2
ADC_CLOCK_4	ADC_INIT_CLK4	clock Divide 1/4
ADC_CLOCK_8	ADC_INIT_CLK8	clock Divide 1/8
ADC_CLOCK_16	ADC_INIT_CLK16	clock Divide 1/16
ADC_CLOCK_32	ADC_INIT_CLK32	clock Divide 1/32
ADC_CLOCK_64	ADC_INIT_CLK64	clock Divide 1/64
ADC_CLOCK_128	ADC_INIT_CLK128	clock Divide 1/128
ADC_CLOCK_256	ADC_INIT_CLK256	clock Divide 1/256
ADC_CH0	ADC_INIT_CH0	Channel 0
ADC_CH1	ADC_INIT_CH1	Channel 1
ADC_CH2	ADC_INIT_CH2	Channel 2
ADC_CH3	ADC_INIT_CH3	Channel 3
ADC_CH4	ADC_INIT_CH4	Channel 4
ADC_CH5	ADC_INIT_CH5	Channel 5
ADC_CH6	ADC_INIT_CH6	Channel 6
ADC_CH7	ADC_INIT_CH7	Channel 7
ADC_NORMAL	0x0000	mode Select Normal
ADC_CONTINUE	ADC_TRIG_CHNL_MS	mode Select Continue
ADC_EXTERN	0x0018	trigger Select AD Trigger
ADC_TIMER_8	0x0010	trigger Select 8bit Timer
ADC_TIMER_16	0x0008	trigger Select 16bit Timer
ADC_SOFTWARE	0x0000	trigger Select Software
ADC_INT_CMPONLY	ADC_EN_SMPL_STAT_INTMODE	intSignalMode Complete only
ADC_INT_OR	0x0000	intSignalMode OR mode
ADC_SAMPLING_9	0x0300	sampling Time Clock9
ADC_SAMPLING_7	0x0200	sampling Time Clock7
ADC_SAMPLING_5	0x0100	sampling Time Clock5
ADC_SAMPLING_3	0x0000	sampling Time Clock3
ADC_INT_CH0	ADC_CH07_INTMASK_INTMASK0	Channel 0 interrupt
ADC_INT_CH1	ADC_CH07_INTMASK_INTMASK1	Channel 1 interrupt
ADC_INT_CH2	ADC_CH07_INTMASK_INTMASK2	Channel 2 interrupt
ADC_INT_CH3	ADC_CH07_INTMASK_INTMASK3	Channel 3 interrupt
ADC_INT_CH4	ADC_CH07_INTMASK_INTMASK4	Channel 4 interrupt
ADC_INT_CH5	ADC_CH07_INTMASK_INTMASK5	Channel 5 interrupt
ADC_INT_CH6	ADC_CH07_INTMASK_INTMASK6	Channel 6 interrupt
ADC_INT_CH7	ADC_CH07_INTMASK_INTMASK7	Channel 7 interrupt
ADC_STATUS_UPPER	ADC_EN_SMPL_STAT_ADUPRST	Out of upper limit value
ADC_STATUS_LOWER	ADC_EN_SMPL_STAT_ADLWRST	Out of lower limit value
ADC_STATUS_CMP	ADC_EN_SMPL_STAT_ADF	Conversion complete
ADC_STATUS_RUN	ADC_EN_SMPL_STAT_ADST	Currently in conversion process
ADC_STATUS_ERR	ADC_EN_SMPL_STAT_OWE	Overwrite error
ADC_STATUS_CH0_CMP	ADC_END_ADF0	Ch.0 Conversion complete
ADC_STATUS_CH1_CMP	ADC_END_ADF1	Ch.1 Conversion complete
ADC_STATUS_CH2_CMP	ADC_END_ADF2	Ch.2 Conversion complete
ADC_STATUS_CH3_CMP	ADC_END_ADF3	Ch.3 Conversion complete
ADC_STATUS_CH4_CMP	ADC_END_ADF4	Ch.4 Conversion complete
ADC_STATUS_CH5_CMP	ADC_END_ADF5	Ch.5 Conversion complete
ADC_STATUS_CH6_CMP	ADC_END_ADF6	Ch.6 Conversion complete
ADC_STATUS_CH7_CMP	ADC_END_ADF7	Ch.7 Conversion complete
ADC_STATUS_CH0_ERR	ADC_END_OWE0	Ch.0 Overwrite error
ADC_STATUS_CH1_ERR	ADC_END_OWE1	Ch.1 Overwrite error
ADC_STATUS_CH2_ERR	ADC_END_OWE2	Ch.2 Overwrite error
ADC_STATUS_CH3_ERR	ADC_END_OWE3	Ch.3 Overwrite error
ADC_STATUS_CH4_ERR	ADC_END_OWE4	Ch.4 Overwrite error
ADC_STATUS_CH5_ERR	ADC_END_OWE5	Ch.5 Overwrite error
ADC_STATUS_CH6_ERR	ADC_END_OWE6	Ch.6 Overwrite error
ADC_STATUS_CH7_ERR	ADC_END_OWE7	Ch.7 Overwrite error
ADC_INTERNAL_ST_IDLE	0x0000	IDLE state
ADC_INTERNAL_ST_SAMP	0x0010	Sampling

4. SOFTWARE DESCRIPTION

Definition name	Value	Description
ADC_INTERNAL_ST_RSV	0x0020	Reserved
ADC_INTERNAL_ST_CONV	0x0030	Currently in conversion process
ADC_INTR_BUFFER_FULL	0x0001	Unused
ADC_INTR_UPPER_OUT	0x0002	Out of upper limit value
ADC_INTR_LOWER_OUT	0x0004	Out of lower limit value
ADC_INTR_CONVERT_CMP	0x0008	Conversion complete
ADC_INTR_OWE	0x0010	Overwrite
ADC_DATA_OWE	0x8000	Conversion result: Overwrite generation flag
ADC_DATA_VALID	0x4000	Conversion result: Data validity flag
ADC_INT_ENABLE	0	Interrupt enable flag
ADC_INT_DISABLE	1	Interrupt disable flag

REVISION HISTORY

AMERICA

EPSON ELECTRONICS AMERICA, INC.

HEADQUARTERS

2580 Orchard Parkway
San Jose , CA 95131,USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

SALES OFFICES

Northeast

301 Edgewater Place, Suite 210
Wakefield, MA 01880, U.S.A.
Phone: +1-800-922-7667 FAX: +1-781-246-5443

EUROPE

EPSON EUROPE ELECTRONICS GmbH

HEADQUARTERS

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

ASIA

EPSON (CHINA) CO., LTD.

7F, Jinbao Bldg., No.89 Jinbao St.,
Dongcheng District,
Beijing 100005, China
Phone: +86-10-6410-6655 FAX: +86-10-6410-7320

SHANGHAI BRANCH

7F, Block B, Hi-Tech Bldg., 900, Yishan Road,
Shanghai 200233, CHINA
Phone: +86-21-5423-5522 FAX: +86-21-5423-5512

EPSON HONG KONG LTD.

20/F., Harbour Centre, 25 Harbour Road
Wanchai, Hong Kong
Phone: +852-2585-4600 FAX: +852-2827-4346
Telex: 65542 EPSCO HX

EPSON (CHINA) CO., LTD.

SHENZHEN BRANCH
12/F, Dawning Mansion, Keji South 12th Road,
Hi-Tech Park, Shenzhen
Phone: +86-755-2699-3828 FAX: +86-755-2699-3838

EPSON TAIWAN TECHNOLOGY & TRADING LTD.

14F, No. 7, Song Ren Road,
Taipei 110
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 CORPORATION

KOREA OFFICE

50F, KLI 63 Bldg., 60 Yoido-dong
Youngdeungpo-Ku, Seoul, 150-763, KOREA
Phone: +82-2-784-6027 FAX: +82-2-767-3677

GUMI OFFICE

2F, Grand B/D, 457-4 Songjeong-dong,
Gumi-City, KOREA
Phone: +82-54-454-6027 FAX: +82-54-454-6093

SEIKO EPSON CORPORATION SEMICONDUCTOR OPERATIONS DIVISION

IC Sales Dept.

IC International Sales Group

421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117