AN211319

Using an ADC in Traveo™ Family S6J3200/S6J3300/S6J3350/S6J3360/S6J3370/ S6J3400/S6J3510 Series

Author: Yoshihiro Tsunokawa

Associated Part Family: Traveo Family S6J3200/3300/3350/3360/3370/3400/3510 Series

Related Documents: For a complete list, see Related Documents.

AN211319 demonstrates how to configure and use an ADC in an S6J3200/ S6J3300/ S6J3350/S6J3360/S6J3370/ S6J3400/S6J3510 series MCU with a software trigger, a hardware trigger, group processing, range compare, pulse detection, diagnosis, and calibration.

Contents

1 Introduction

This application note is intended for users of the Cypress Traveo family S6J3200/S6J3300/S6J3350/S6J3360/ S6J3370/S6J3400/S6J3510 series MCUs. It describes how to use the ADC interface.

Even though the example applications in this document are intended to use S6J3200 series MCUs, you can use another series MCU; see the datasheet and hardware manuals for MCUs that you want to use instead of those for S6J3200 series. The datasheet and hardware manual links are here.

2 Overview of ADC in S6J3200/S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/ S6J3510 Series MCUs

The ADC converts analog input voltages into digital values. It features 8 range comparators, 64 pulse detection units, 64 conversion data registers, and 4 multiple conversion channels.

For details, see the 12/10/8-Bit Analog to Digital Converter chapter in the Hardware Manual.

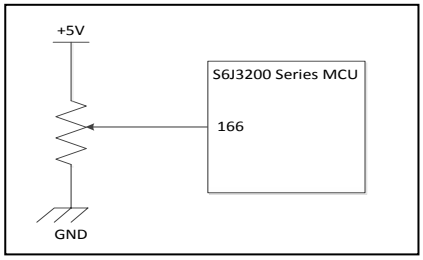
3 Software Trigger Procedure

This section shows an example application that converts voltage values given to the pin number 166 at the S6J3200L Series MCUs to digital values as shown in Figure 1 and stores the values into RAM of the MCU. Analog-to-digital conversion is repeated in an infinite loop in the main function by using software triggers.

All applications in this document are intended to be used with S6J3200L Series MCUs that have 216 pins.

If you intend to use S6J3200K Series MCUs that have 208 pins, please see the column ‘TEQFP208’ instead of ‘TEQFP216’ in Table 1 at the procedure of Port Settings.

Figure 1. Example of Analog-to-Digital Conversion Connection



To implement this application, use the following sections that describe the procedure for setting the ADC channel. They also provide example code for using a software trigger and the example code of the method to obtain the value of the ADC.

3.1 Port Settings

This section describes how to set the port corresponding to pin number 166 for an analog input.

Most ports of S6J3200 series MCUs must be set to a function from General I/O, Peripheral I/O, or ADC input.

Find the pin number 166 from the hardware manual, Chapter 10: “Port Description,” line of “Package Pin Number TEQFP216.” Pin number 166 is allocated to Port3_07 and analog input 15 (AN15) in the 216-pin package, as shown in Table 1.

Analog-to-digital conversion is not possible at all pins. You must select the pin to use for analog input in the beginning itself according to this table.

Table 1. Port Location Table

Port Name	Description	Package Pin Number		Remark
		TEQFP208	TEQFP216	
AN10	ADC Analog 10 input pin	98	102	
AN11	ADC Analog 11 input pin	99	103	
AN12	ADC Analog 12 input pin	100	104	
AN13	ADC Analog 13 input pin	101	105	
AN14	ADC Analog 14 input pin	102	106	
AN15	ADC Analog 15 input pin	160	166	
AN16	ADC Analog 16 input pin	161	167	
P3_03	General-Purpose I/O port	99	103	
P3_04	General-Purpose I/O port	100	104	
P3_05	General-Purpose I/O port	101	105	
P3_06	General-Purpose I/O port	102	106	
P3_07	General-Purpose I/O port	160	166	
P3_08	General-Purpose I/O port	161	167	
P3_09	General-Purpose I/O port	162	168	

The example code of setting Port3_07 to analog input follows (Code 1).

All example codes in this document are based on the premise that CLK_LCP1A, which is provided to the ADC, is set to 120 MHz.

Code 1. Example Code of Port Settings

```
/********** Port Settings For AN15 **********/
PPC_KEYCDR = 0x100000CE;
PPC_KEYCDR = 0x500000CE;
PPC_KEYCDR = 0x900000CE;
PPC_KEYCDR = 0xD00000CE;
PPC_PCFGR307 = 0x0000;

GPIO_KEYCDR = 0x2000003C;
GPIO_KEYCDR = 0x6000003C;
GPIO_KEYCDR = 0xA000003C;
GPIO_KEYCDR = 0xE000003C;
GPIO_DDCR3 = 0x00000080;
```

Key code register operation

Port setting for analog input

Key code register operation

Port setting for analog input

3.1.1 Code Description

The Traveo Family Hardware Manual specifies that the PIE bit, POF bit, PDE bit, PUE bit of the PPC_PCFGR register, and the GPIO_DDR register of the target port should be set to ‘0’ for analog input. See the *S6J3200, HARDWARE MANUAL*, Chapter 11, section 3.3, “Analog I/O Setting”.

The other bits are set to ‘0’ because these bits do not affect the analog input.

3.1.2 Key Code Register Operation

The functions of the PPC_PCFGR307 and GPIO_DDCR3 registers are set by a key code register. See *S6J3200, Traveo Family Hardware Manual Platform Part*, Chapter 49, section 4.9, “GPIO Key Code Register (GPIO_KEYCDR)” and section 4.11, “PPC Key Code Register (PPC_KEYCDR)” for details of the key code register setting.

The key code register value depends on the address of the target register.

3.1.3 Port input Enable Bit Setting

PPC_PCFGR307 is the register for the settings of P3_07, which corresponds to AN15. See the *S6J3200, Traveo Family Hardware Manual Platform Part*, Chapter 49, section 4.10, “Port Setting Register (PPC_PCFGRijj)” for more information.

GPIO_DDCR3 is the register for clearing the GPIO_DDR register. See the *S6J3200, Traveo Family Hardware Manual Platform Part*, Chapter 49, section 4.3, “Data Direction Clear Register (GPIO_DDCRi)” for more information.

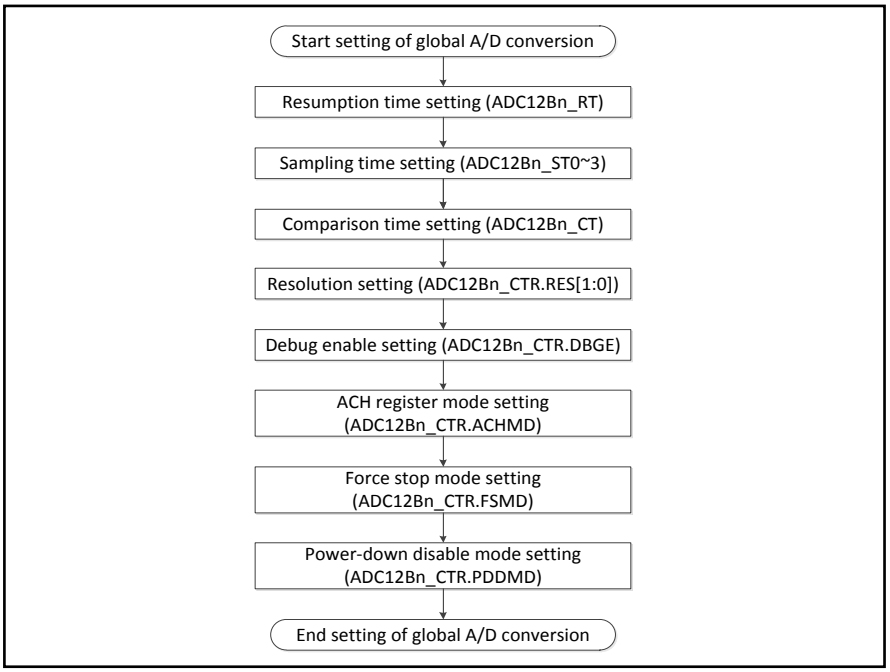
PPC_PCFGR307 and GPIO_DDR3 are set to ‘0’ in this example code.

3.2 ADC Global Settings

This section provides the procedure to configure common ADC settings in each channel and its example code.

Process the ADC global settings in accordance with the flow chart (Figure 2).

Figure 2. ADC Global Settings Flow Chart



The example code of the ADC global settings follows (Code 2).

All sample codes in this document are based on the assumption that CLK_LCP1A which is provided to A/D Converter is set to 120 MHz.

Code 2. Example Code of ADC Global Settings

```
/********** A/D Converter Global Settings **********/
ADC12B0_RT = 120;          <-- Resumption time setting
ADC12B0_ST0 = 36;
ADC12B0_ST1 = 36;          <-- Sampling time setting
ADC12B0_ST2 = 36;
ADC12B0_ST3 = 36;
ADC12B0_CT = 8;            <-- Comparison time setting
ADC12B0_CTRL_RES = 0;      <-- Resolution setting
ADC12B0_CTRL_DBGE = 0;     <-- Debug enable setting
ADC12B0_CTRL_ACHMD = 0;    <-- ACH register mode setting
ADC12B0_CTRL_FSMD = 0;     <-- Force stop mode setting
ADC12B0_CTRL_PDDMD = 1;    <-- Power-down disable mode setting
```

3.2.1 Code Description

The clock provided to ADC is CLK_LCP1A.

3.2.2 Resumption Time Setting

Resumption time is calculated using the following equation:

$$\text{Resumption time}[s] = \text{ADC12B0\_RT}/\text{CLK\_LCP1A}[\text{Hz}] \quad \text{Equation 1}$$

To find out the maximum value of the resumption time, refer to the *S6J3200 Series, Datasheet* (see Table 2).

The current maximum value is 1 µs.

It is recommended that you set the resumption time to the minimum value within the range of the specification.

Considering this recommendation and that CLK_LCP1A is 120 MHz and in accordance with Equation (1),

$$1[\mu s] \leq \text{ADC12B0\_RT}/120[\text{MHz}]$$

According to this formula,

$$\text{ADC12B0\_RT} \geq 120$$

This shows that the ADC12B0_RT value should be set to 120 units.

3.2.3 Sampling Time Setting

Sampling time is calculated using the following equation:

$$\text{Sampling time}[s] = \text{ADC12B0\_STx}/\text{CLK\_LCP1A}[\text{Hz}] \quad \text{Equation 2}$$

To find out the minimum value of the sampling time, refer to the *S6J3200 Series, Datasheet* (see Table 2).

The current minimum value is 0.3 µs.

It is recommended that you set the sampling time to the minimum value within the range of the specification.

Considering this recommendation and that CLK_LCP1A is 120 MHz and in accordance with Equation (2),

$$0.3[\mu s] \leq \mathrm{ADC12B0\_STx}/120[\mathrm{MHz}]$$

According to this formula,

$$\mathrm{ADC12B0\_STx} \geq 36$$

This shows that the ADC12B0_STx value should be set to 36 units.

In the global settings procedure, the sampling time can be set as four types. In the logical channel setting procedure, you can specify which of the four is used.

3.2.4 Comparison Time Setting

Comparison time is calculated using the following equation.

If [ADC12B0_CT value < 4]:

$$\mathrm{Compare\ time}[s] = (\mathrm{ADC12B0\_CT} \times 12 + 4)/\mathrm{CLK\_LCP1A}[\mathrm{Hz}] \quad \text{Equation 3}$$

If [ADC12B0_CT value >= 4]:

$$\mathrm{Compare\ time}[s] = (\mathrm{ADC12B0\_CT} \times 13)/\mathrm{CLK\_LCP1A}[\mathrm{Hz}] \quad \text{Equation 4}$$

To find out the minimum value of the sampling time, refer to the *S6J3200 Series, Datasheet* (see Table 2).

The current minimum value is 0.8 µs.

It is recommended that you set the comparison time to the minimum value within the range of the specification.

Considering this recommendation and that CLK_LCP1A is 120 MHz and in accordance with Equations (3) and (4),

$$0.8[\mu s] \leq (\mathrm{ADC12B0\_CT} \times 13)/120[\mathrm{MHz}]$$

According to this formula,

$$\mathrm{ADC12B0\_CT} \geq 7.38$$

This shows that the ADC12B0_CT value should be set to 8 units.

Table 2. Sampling, Compare, and Resumption Time from ADC Device Datasheet

Parameter	Symbol	Pin Name	Value			Unit	Remarks
			Min	Typ	Max		
Resolution	-	-	-	-	12	bit	
Total Error	-	-	-	-	±12	LSB	
Integral Non linearity	-	-	-	-	±4.0	LSB	
Differential Non linearity	-	-	-	-	±1.9	LSB	
Zero transition voltage	V_{ZT}	AN0 to AN49	AVRL -11.5LSB	-	AVRL +12.5LSB	V	
Full-scale transition voltage	V_{FST}	AN0 to AN49	AVRH -13.5LSB	-	AVRH +10.5LSB	V	
Sampling time	t_{SMP}	-	0.3	-	-	µs	
Compare time	t_{CMP}	-	0.8	-	28	µs	
A/D conversion time	t_{CNV}	-	1.1	-	-	µs	
A/D trigger input time		ADTRG	$4t_{CLK\_LCP1A}$	-	-	ns	$4t_{CLK\_LCP1A} \geq$ 100ns
			100				$4t_{CLK\_LCP1A} <$ 100ns
Resumption time	-	-	-	-	1	us	-
Analog port input current	I_{AIN}	AN0 to AN17	-1.0	-	1.0	µA	$V_{AVSS} \leq - V_{AIN} \leq V_{AVCC}$
		AN18 to AN25	-2.0	-	2.0	µA	
		AN26 to AN49	-3.0	-	3.0	µA	
Analog input voltage	V_{AIN}	AN0 to AN49	AVSS	-	AVRH	V	
Reference voltage	AVRH	AVRH5	4.5	-	5.5	V	$AV_{CC} \geq$ AVRH
	AVRL	AVRL5/AVSS	-	0.0	-	V	
Power supply current	I_A	AVCC	-	500	900	µA	
	I_{AH}		-	1.0	100	µA	
	I_R	AVRH	-	1.0	2.0	mA	
	I_{RH}		-	-	5.0	µA	
Variation between channels	-	AN0 to AN49	-	-	4.0	LSB	

3.2.5 Resolution Setting

The resolution of the ADC is selected as 8-bit, 10-bit, or 12-bit.

ADC12B0_CTRL_RES = 0 or 2: 12-bit mode

ADC12B0_CTRL_RES = 1: 10-bit mode

ADC12B0_CTRL_RES = 3: 8-bit mode

12-bit mode is selected in this example code.

3.2.6 Debug Enable Setting

When this bit is set to ‘1’ and the processor is in the debug state, the ADC completes the current conversion, but further conversion is stopped.

When the processor leaves the debug state or DBGE is set to ‘0’, conversion continues with the next channel from where it had stopped. It is set to disable in this example code.

3.2.7 ACH Register Mode Setting

Select direct ACH register mode or latched ACH register mode.

ADC12B0_CTRL_ACHMD = 0: Direct ACH register mode

ADC12B0_CTRL_ACHMD = 1: Latched ACH register mode

In direct ACH register mode, ADC12Bn_STAT.ACH shows the number of currently converted logical channels.

In latched ACH register mode, ADC12Bn_STAT.ACH shows the number of the logical channel whose conversion was finished last.

3.2.8 Force Stop Mode Setting

Select whether force stop mode is use.

ADC12B0_CTRL_FSMD = 1: Force stop mode

ADC12B0_CTRL_FSMD = 0: Not force stop mode

See the *S6J3200, Hardware Manual*, section 15.3.7 to learn about force stop mode.

It is set to not force stop mode in this example code.

3.2.9 Power-Down Disable Mode Setting

Select whether power-down disable mode is used.

ADC12B0_CTRL_PDDMD = 1: Power-down disable mode

ADC12B0_CTRL_PPDMD = 0: Not power-down disable mode

See the *S6J3200, Hardware Manual*, section 15.3.1 to learn about power-down disable mode.

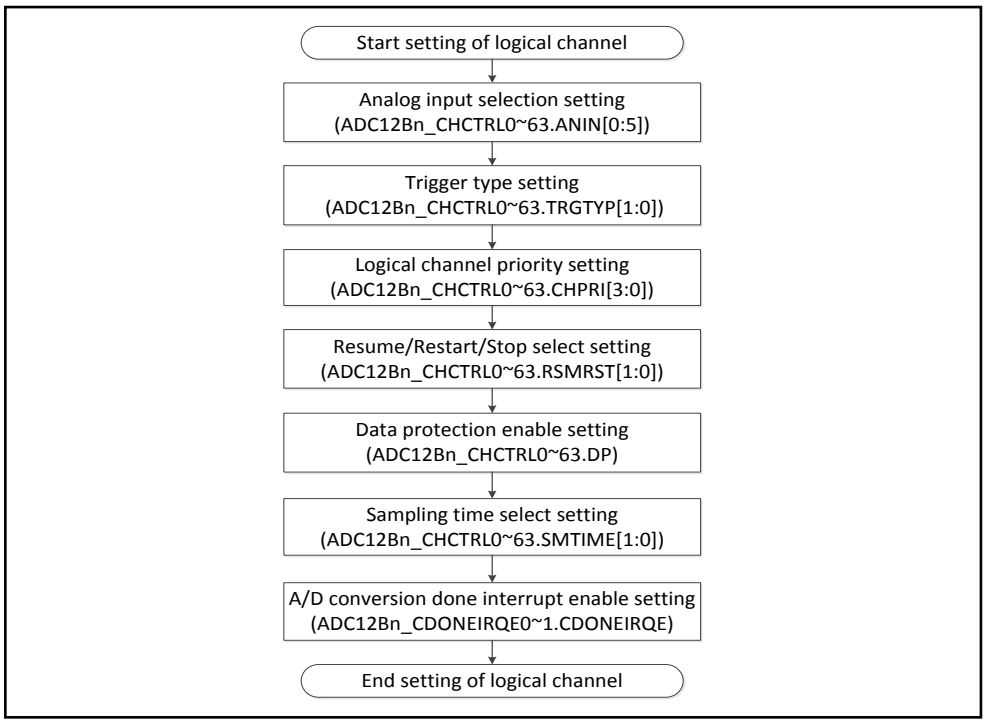
It is set to power-down disable mode in this example code.

3.3 Logical Channel Settings with Software Trigger

There is one A/D Conversion Data Register per logical channel. A logical channel can be assigned any analog input (AN) by setting the register ADC12Bn_CHCTRLx_ANIN.

Figure 3 shows the flow chart of the logical channel setting with a software trigger.

Figure 3. Logical Channel Setting with Software Trigger Flow Chart



The example code of the ADC logical channel settings with a software trigger is given in Code 3.

Logical channel 0 is used in this example code. Any logical channel will work in this application as long as proper analog input is selected.

Code 3. Example Code of ADC Logical Channel Settings with Software Trigger

```
/********** A/D Converter Ch0 Settings **********/
ADC12B0_CHCTRL0_ANIN = 15;                    <-- Analog input selection setting
ADC12B0_CHCTRL0_TRGTYP = 0;                   <-- Trigger type setting
ADC12B0_CHCTRL0_CHPRI = 0;                    <-- Logical channel priority setting
ADC12B0_CHCTRL0_RSMRST = 1;                   <-- Resume/restart/stop select setting
ADC12B0_CHCTRL0_DP = 0;                       <-- Enable data protection
ADC12B0_CHCTRL0_SMTIME = 0;                   <-- Sampling time select setting
ADC12B0_CDONEIRQE0_CDONEIRQE0 = 0;            <-- Conversion done interrupt enable setting
ADC12B0_GRPIRQE0_GRPIRQE0 = 0;                <-- Group interrupted interrupt enable setting
```

3.3.1 Analog Input Selection Setting

Pin number 166 corresponds to analog input 15 (AN15).

Allocate ADC logical channel 0 to analog input 15 (AN15).

3.3.2 Trigger Type Setting

Set the trigger type. The trigger type depends on the TRGTYP register as follows.

TRGTYP = 0b00: Software trigger only

TRGTYP = 0b01: Software or hardware trigger

TRGTYP = 0b10: Conversion completion trigger

TRGTYP = 0b11: Idle trigger

It is set to '0' in this example code (software trigger only).

3.3.3 Logical Channel Priority Setting

Set the logical channel priority.

Setting the register to '0' corresponds to the highest priority and '15' corresponds to the lowest priority.

It is set to '0' in this example code (highest priority).

3.3.4 Resume/Restart/Stop Select Setting

The RSMRST register defines how the group processing is to be continued after the execution of a higher-priority conversion finishes.

RSMRST = 0: Stop

RSMRST = 1: Resume

RSMRST = 2: Restart

It is set to '1' (Resume) in this example code. However, this setting has no effect on the ADC operation because only one channel is used.

3.3.5 Enable Data Protection

If this bit is equal to '1', the data protection function is enabled.

If the data protection function is enabled, the next conversion of this logical channel is not started until the previous data has been read out.

In this example code, it is set to '0'. This function is not used.

3.3.6 Sampling Time Select Setting

Select the sampling time to use this logical channel from the ADC12B0_STx(x =0~3) setting in the global setting procedure.

It is selected as ADC12B0_ST0 in this example code.

3.3.7 Conversion Done Interrupt Enable Setting

0: Conversion done interrupt disabled

1: Conversion done interrupt enabled

Conversion done interrupt is issued when the bit is '1' and the corresponding interrupt flags ADC12Bn_CDONEIRQ1.CDONEIRQ63 to 32 and ADC12Bn_CHSTAT32 to 63.CDONEIRQ are set to '1'.

In this example code, it is set to '0'. This function is not used.

3.3.8 Group Interrupted Interrupt Enable Setting

0: Group interrupted interrupt disabled

1: Group interrupted interrupt enabled

A group interrupted interrupt is issued when this bit is '1' and the corresponding interrupt flags

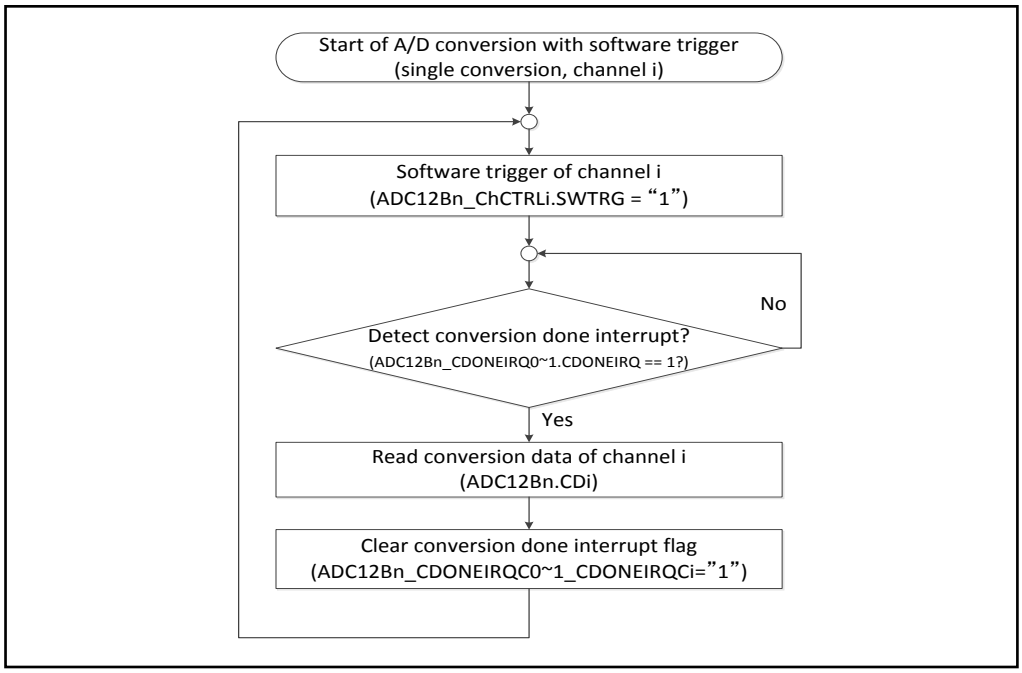
(ADC12Bn_GRPIRQ0.GRPIRQ31 to 0 and ADC12Bn_CHSTAT0 to 31.GRPIRQ) are set to '1'.

In this example code, it is set to '0'. This function is not used.

3.4 Analog-to-Digital Conversion with Software Trigger

Figure 4 shows the flow chart of analog-to-digital conversion with a software trigger.

Figure 4. Analog-to-Digital Conversion with Software Trigger Flow Chart



The example code of analog-to-digital conversion with a software trigger is given in Code 4.

Code 4. Example Code of Analog-to-Digital Conversion with Software Trigger

```
/********** Endless main loop **********/
for (;;)
{
    /********** Software Trigger **********/
    ADC12B0_CHCTRL0_SWTRG = 1;          <-- Software trigger of channel 0
    /********** Wait A/D Convert Completion **********/
    while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)   <-- Detect conversion done interrupt
    {
        ClearWatchdog();
    }
    /********** Get AD Value **********/
    Volume = ADC12B0_CD0;               <-- Read conversion data of channel 0
    /********** Clear Conversion Interrupt Flag **********/
    ADC12B0_CDONEIRQC0_CDONEIRQC0 = 1;  <-- Clear conversion done interrupt flag
    /********** Clear hardware watchdog **********/
    ClearWatchdog();
```

```
    }
```

3.4.1 Software Trigger of Channel 0

Release the software trigger of ADC channel 0 by writing ‘1’ to ADC12B0_CHCTRL0_SWTRG.

3.4.2 Detect Conversion Done Interrupt

Discover whether the ADC finished conversion by detecting the conversion done interrupt flag.

If the flag has been set, quit the ‘while’ loop.

3.4.3 Read Conversion Data of Channel 0

Read the conversion data to the variable ‘Volume.’

3.4.4 Clear Conversion Done Interrupt Flag

Clear the ‘conversion done’ interrupt flag by writing ‘1’ to ADC12B0_CDONEIRQ0_CDONEIRQ0.

4 Hardware Trigger Procedure

The ADC in S6J3200 can be triggered by other related hardware functions, such as reload timer, output compare, and base timer.

This section shows an example application that converts voltage values given to the pin number 166 at the MCU to digital values and stores the values into the RAM of the MCU. The analog-to-digital conversion is repeated at regular timing by hardware trigger of a reload timer.

The physical setting of this application is the same as shown in Figure 1 in Software Trigger Procedure.

Ensure that you have set up the parameters per section Port Settings and Global Settings.

To implement this application, follow the procedure of ADC channel setting and its example code when using a hardware trigger. The example in this section uses reload Timer ch0 for the trigger.

4.1 Logical Channel Setting with Hardware Trigger

The flowchart of the logical channel settings with a hardware trigger is the same as that shown in Figure 3.

An example code of the logical channel settings with a hardware trigger is given in Code 5.

Code 5. Example Code of Logical Channel Setting with Hardware Trigger

```
/********** A/D Converter Ch0 Settings **********/
ADC12B0_CHCTRL0_ANIN = 15;
ADC12B0_CHCTRL0_TRGTYP = 1;      <-- Trigger type setting
ADC12B0_CHCTRL0_CHPRI = 0;
ADC12B0_CHCTRL0_RSMRST = 1;
ADC12B0_CHCTRL0_DP = 0;
ADC12B0_CHCTRL0_SMTIME = 0;
ADC12B0_CDONEIRQE0_CDONEIRQE0 = 0;
ADC12B0_GRPIRQE0_GRPIRQE0 = 0;
```

4.1.1 Trigger Type Setting

Set the trigger type. The trigger type depends on the TRGTYP register as follows.

TRGTYP = 0b00: Software trigger only

TRGTYP = 0b01: Software or hardware trigger

TRGTYP = 0b10: Conversion completion trigger

TRGTYP = 0b11: Idle trigger

It is set to '1' (software or hardware trigger) in this example code.

The other settings are set in the same way as noted in the Software Trigger Procedure section.

The other portion of this code is the same as the code given for the software trigger.

4.1.2 Assign Underflow of Reload Timer Channel 0 to Trigger of ADC Channel 0

See the *S6J3200, Hardware Manual*, Chapter 11, Port Configuration, 3.1 "Resource Input Configuration Module" and search for "ADC12B_HWTRG0" in the "Resource" category, as shown in Table 3.

Table 3. Resource Input Table

Register (Offset)	Resource	RESSEL [3:0] /PORTSEL[3:0]	Source for Resource Input							
			0	1	2	3	4	5	6	7
			8	9	10	11	12	13	14	15
RIC_RESIN438 (0x036C)	ADC12B_HWTRG0	RESSEL (0-7)	PORT_PIN	RLT0_UFSET	RLT1_UFSET	OCU0_OTD0	OCU1_OTD0	BT0_ADTOUT0	BT1_ADTOUT2	BT3_ADTOUT2
		RESSEL (8-15)	-	-	-	-	-	-	-	-
		PORTSEL (0-7)	-	-	-	-	-	-	-	-
		PORTSEL (8-15)	-	-	-	-	-	-	-	-

This table shows that RIC_RESIN438 must be set to '1' to assign the underflow of reload timer ch0 to trigger ADC channel 0.

Code 6. Example Code of the ADC Hardware Trigger Assignment

```
/********** Assignment RLT0 to A/D Converter Ch0 HW Trigger **********/
RIC_KEYCDR = 0x1000036C;
RIC_KEYCDR = 0x5000036C;
RIC_KEYCDR = 0x9000036C;
RIC_KEYCDR = 0xD000036C;
RIC_RESIN438 = 0x0001;
```

Key code resister operation

Assign ADC12B_HWTRG0 to reload timer ch0 underflow

4.1.3 Code Description

Resource Input Setting Register RIC_RESIN438 is set to '1'.

RIC_RESINn is the key code register. See the *S6J3200, Traveo Family Hardware Manual Platform Part*, Chapter 49, Section 4.13 for the procedure to set the key code register.

The value that must be set to key code register depends on the address of the target register.

4.1.4 Start Relevant Reload Timer

Set and start reload timer channel 0, which is assigned as an ADC trigger.

The example code of the reload timer start is given in Code 7.

Code 7. Example Code of Reload Timer Start

```
static void Rlt0_Start(void)
{
    /********** Reload Timer 0 Setting **********/
    RLT0_TMRLR = 14999;
    RLT0_TMCSR_CNTE = 1;
    RLT0_TMCSR_CSL = 2;
    RLT0_TMCSR_RELD = 1;
    RLT0_TMCSR_TRG = 1;
}
```

See the *S6J3200, Traveo Family Hardware Manual Platform Part*, Chapter 44 to learn about the reload timer settings.

With this setting, the ADC is conducted at intervals of the reload timer channel0 underflow.

4.2 Analog-to-Digital Conversion with Hardware Trigger

The example code of analog-to-digital conversion with a hardware trigger is given in Code 8.

Code 8. Example Code of Analog-to-Digital Conversion with Hardware Trigger

```
  /********** Endless main loop **********/
    for (;;)
    {
        /********** Wait A/D Convert Completion **********/
        while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)
        {
            ClearWatchdog();
        }
        /********** Get ADC Value **********/
        Volume = ADC12B0_CD0;
        /********** Clear Conversion Interrupt Flag **********/
        ADC12B0_CDONEIRQC0_CDONEIRQC0 = 1;
        /********** Clear hardware watchdog **********/
        ClearWatchdog();
    }
```

4.2.1 Code Description

This code is the same function as the code with the software trigger, except that there is no need for a software trigger.

5 Group Procedure

Though an ADC on the S6J3200 series MCU can't process analog-to-digital conversion using multiple pins at the same time, it has a function for consecutive conversion using multiple pins with one trigger. The pins and the order of conversions can be selected from any ports that can be configured as analog input.

ADC logical channels that are selected for consecutive conversion with one trigger are called 'groups'.

A group is defined by the trigger type configuration of consecutive logical channels. The first channel of the group has the trigger type set to 'software trigger only', 'hardware or software trigger', or 'idle trigger'.

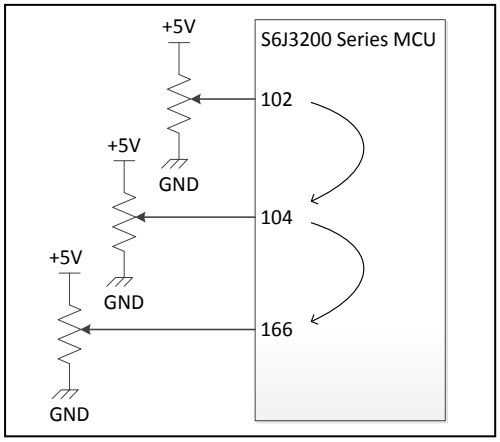
If the following channel trigger type is not set to 'conversion completion trigger', the group consists of only one channel. Otherwise, the group continues until the last channel in a row with the trigger type set to 'conversion completion trigger'.

After the first channel of the group is triggered and converted, the second channel is automatically triggered and so on until the whole group is converted.

This section shows an example application that converts the voltage consecutively in the order of pin 102, pin 104, pin 166 with one software trigger as shown in Figure 5 and stores the digital values into the RAM.

Ensure that you have made the settings described in the Port Settings Global Settings sections.

Figure 5. Example of Group Conversion Connection



Use the information in the following sections and the sample code to implement this application.

5.1 Logical Channel Settings for Group Procedure

The flow chart of each logical channel setting for a group procedure is the same as that with a software trigger.

As shown in Table 1, that pin 102 corresponds to analog input 10 (AN10), pin 104 corresponds to analog input 12 (AN12), and pin 166 corresponds to analog input 15 (AN15).

See Code 9 for example code of the ADC logical channel settings for the group procedure.

Though this example uses logical channels 0, 1, and 2, you can use any channel as long as those numbers are consecutive.

Code 9. Example Code of the ADC Logical Channel Setting for Group Procedure

```
/********** A/D Converter Ch0 Settings **********/
 ADC12B0_CHCTRL0_ANIN = 10;      <-- Analog input selection setting
 ADC12B0_CHCTRL0_TRGTYP = 0;     <-- Set trigger type as software trigger
 ADC12B0_CHCTRL0_CHPRI = 0;
 ADC12B0_CHCTRL0_RSMRST = 1;
 ADC12B0_CHCTRL0_DP = 0;
 ADC12B0_CHCTRL0_SMTIME = 0;
 ADC12B0_CDONEIRQE0_CDONEIRQE0 = 0;
 ADC12B0_GRPIRQE0_GRPIRQE0 = 0;

 /********** A/D Converter Ch1 Settings **********/
 ADC12B0_CHCTRL1_ANIN = 12;      <-- Analog input selection setting
 ADC12B0_CHCTRL1_TRGTYP = 2;     <-- Set trigger type as conversion completion trigger
 ADC12B0_CHCTRL1_CHPRI = 0;
 ADC12B0_CHCTRL1_RSMRST = 1;
 ADC12B0_CHCTRL1_DP = 0;
 ADC12B0_CHCTRL1_SMTIME = 0;
 ADC12B0_CDONEIRQE0_CDONEIRQE1 = 0;
 ADC12B0_GRPIRQE0_GRPIRQE1 = 0;

 /********** A/D Converter Ch2 Settings **********/
 ADC12B0_CHCTRL2_ANIN = 15;      <-- Analog input selection setting
 ADC12B0_CHCTRL2_TRGTYP = 2;     <-- Set trigger type as conversion completion trigger
 ADC12B0_CHCTRL2_CHPRI = 0;
 ADC12B0_CHCTRL2_RSMRST = 1;
 ADC12B0_CHCTRL2_DP = 0;
 ADC12B0_CHCTRL2_SMTIME = 0;
 ADC12B0_CDONEIRQE0_CDONEIRQE2 = 0;
 ADC12B0_GRPIRQE0_GRPIRQE2 = 0;
```

5.1.1 Analog Input Selection Setting

Set the analog input pin that you want to put into a group.

The analog input of logical channels are set to pin 102 (AN10), pin 104 (AN12), and pin 166 (AN15) in this example code.

5.1.2 Trigger Type Setting

Set the trigger type of the ADC logical channel that has the smallest number in the group to the type you want to use, except for ‘conversion completion trigger’, which is set to ‘0’ (software trigger) in this example code.

Set the trigger type of every other consecutive ADC logical channel to ‘2’ (conversion completion trigger). This causes the completion of ADC logical channel 0 to trigger ADC logical channel 1, and the completion of ADC logical channel 1 to trigger ADC logical channel 2.

5.2 Analog-to-Digital Conversion for Group Procedure

See Code 10 for example code of analog-to-digital conversion for the group procedure.

Code 10. Example Code of Analog-to-Digital Conversion for Group Procedure

```
/********** Endless main loop **********/
 for(;;)
 {
     /********** Software Trigger **********/
     ADC12B0_CHCTRL0_SWTRG = 1;
     /********** Wait A/D Convert Completion **********/
     while(ADC12B0_STAT_BUSY)        <-- Detect ADC busy flag
     {
         ClearWatchdog();
     }
     /********** Get ADC Value **********/
     Volume0 = ADC12B0_CD0;
     Volume1 = ADC12B0_CD1;
     Volume2 = ADC12B0_CD2;
     /********** Clear hardware watchdog **********/
     ClearWatchdog();
 }
```

5.2.1 Detect ADC Busy Flag

In the group procedure, ADC completions are confirmed by the detection of the ADC busy flag.

After all conversion requests are processed, BUSY flag becomes 0 until next conversion request appears and resumption time elapses.

6 Range Comparator Procedure

The range comparator offers eight comparison groups, each with an upper and lower threshold register.

The 64 logical channels can be enabled for range comparison and assigned to one of the eight comparators individually.

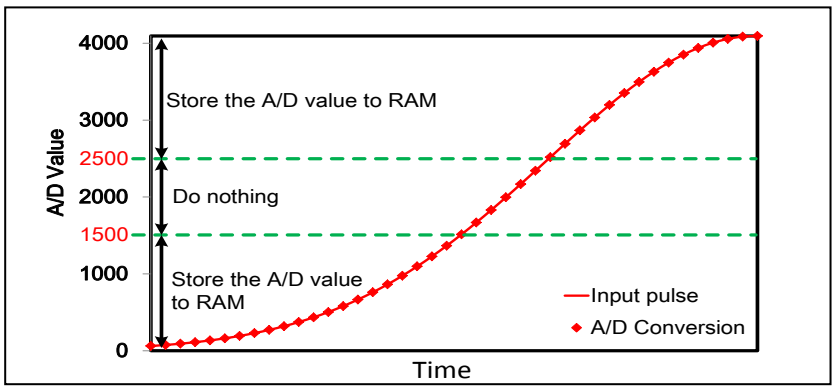
This function is usually used to monitor abnormal values in the voltage.

This section shows an example application that converts voltage values given to the pin number 166 at the MCU to digital values. If the digital value is larger than ‘2500’ or smaller than ‘1500’, it stores the value into the RAM of the MCU as shown in Figure 6. Analog-to-digital conversion is repeated in an infinite loop in the main function using software trigger.

The physical setting of this application is the same as shown in Figure 1.

Ensure that you have configured the settings as described in the Port Settings and ADC Global Settings sections.

Figure 6. Behavior of Range Comparator application

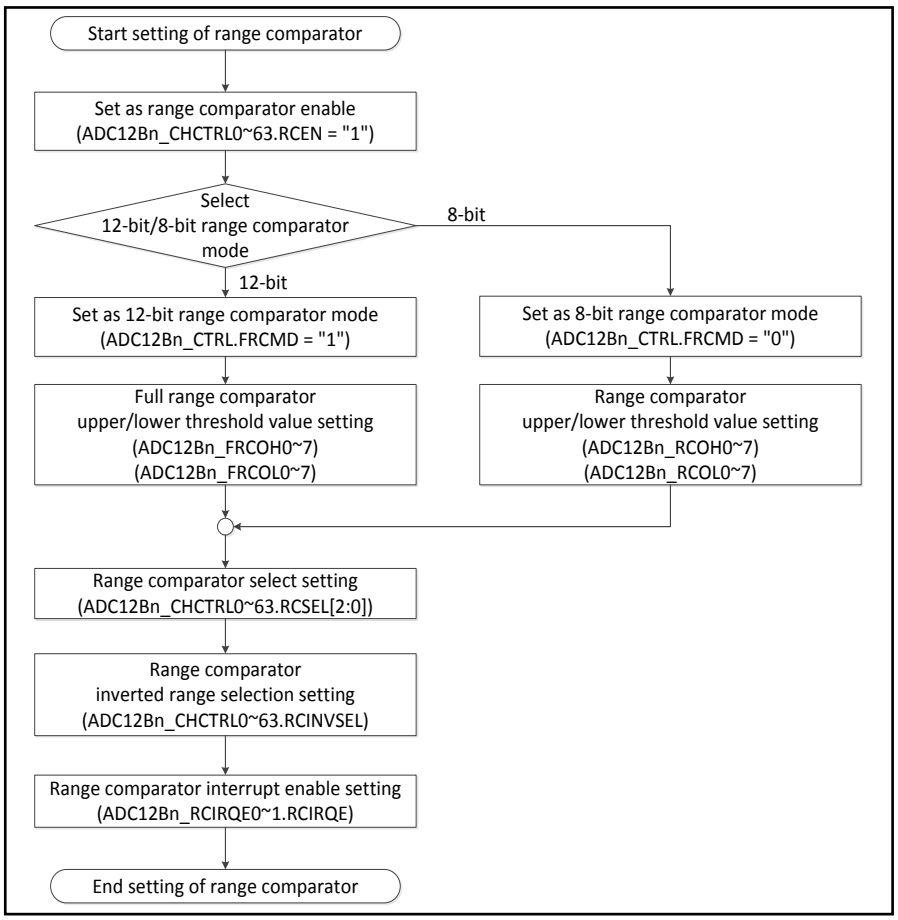


Use the information in the following sections and the example code to implement this application.

6.1 Range Comparator Settings

Figure 7 shows the range comparator setting flow chart.

Figure 7. Range Comparator Setting Flow Chart



See Code 11 for the example code of range comparator settings.

Code 11. Example Code of Range Comparator Settings

```
/********** A/D Converter Range Global Settings **********/
 /********** A/D Converter Range Ch0 Settings **********/
 ADC12B0_CHCTRL0_RCEN = 1;          <- Set as range comparator enable
 ADC12B0_CTRL_FRCMD = 1;            <- Set as 12-bit range comparator mode
 ADC12B0_FRCOH0 = 2500;
 ADC12B0_FRCOL0 = 1500;
 ADC12B0_FRCOH1 = 4095;
 ADC12B0_FRCOL1 = 0;
 ADC12B0_FRCOH2 = 4095;
 ADC12B0_FRCOL2 = 0;
 ADC12B0_FRCOH3 = 4095;
 ADC12B0_FRCOL3 = 0;                <- Full-range comparator upper/lower
 ADC12B0_FRCOH4 = 4095;                threshold value setting
 ADC12B0_FRCOL4 = 0;
 ADC12B0_FRCOH5 = 4095;
 ADC12B0_FRCOL5 = 0;
 ADC12B0_FRCOH6 = 4095;
 ADC12B0_FRCOL6 = 0;
 ADC12B0_FRCOH7 = 4095;
 ADC12B0_FRCOL7 = 0;
 ADC12B0_CHCTRL0_RCSEL = 0;         <- Range comparator select setting
 ADC12B0_CHCTRL0_RCINVSEL = 0;      <- Range comparator inverted range select setting
 ADC12B0_RCIRQE0_RCIRQE0 = 0;       <- Range comparator interrupt enable setting
```

6.1.1 Set Range Comparator Enable

ADC12B0_CHCTRL0_RCEN is the range comparator enable bit.

0: Range comparator disabled

1: Range comparator enabled

6.1.2 Set 12-bit/8-bit Range Comparator Mode

ADC12B0_CTRL_FRCMD is the full range comparator mode select bit.

0: 8bit-range comparator mode

1: 12bit-range comparator mode

In the example code, it is set to ‘1’ (12-bit-range comparator mode).

6.1.3 Full-Range Comparator Upper/Lower Threshold Value Setting

The ADC12B0_FRCOHx registers specify the upper threshold values that can be selected for the 12-bit range comparator to which the output of the ADC is compared while.

ADC12B0_FRCOLx registers specify the lower threshold values that can be selected for the 12-bit range comparator to which the output of the ADC is compared.

In the 12-bit-range comparator mode (ADC12Bn_CTRL.FRCMD = "1"), these registers are used for the 12-bit range comparator.

In the 8-bit-range comparator mode (ADC12Bn_CTRL.FRCMD = "0"), the ADC12B0_RCOHx registers are used instead of these registers.

In the example code, FRCOH0 is set to ‘2500’ and FRCOL0 is set to ‘1500’. Other FRCOHx are set to ‘4095’, and FRCOLx are set to ‘0’. These are not used.

6.1.4 Range Comparator Select Setting

RCSEL is the range comparator threshold value select bit.

000: Select range comparator 0, defined by the ADC12B0_FRCOH0 and ADC12B0_FRCOL0 registers or the ADC12B0_RCOH0 and ADC12B0_RCOL0 registers.

001: Select range comparator 1, defined by the ADC12B0_FRCOH1 and ADC12B0_FRCOL1 registers or the ADC12B0_RCOH1 and ADC12B0_RCOL1 registers.

010: Select range comparator 2, defined by the ADC12B0_FRCOH2 and ADC12B0_FRCOL2 registers or the ADC12B0_RCOH2 and ADC12B0_RCOL2 registers.

011: Select range comparator 3, defined by the ADC12B0_FRCOH3 and ADC12B0_FRCOL3 registers or the ADC12B0_RCOH3 and ADC12B0_RCOL3 registers.

⋮

111: Select range comparator 7, defined by the ADC12B0_FRCOH7 and ADC12B0_FRCOL7 registers or the ADC12B0_RCOH7 and ADC12B0_RCOL7 registers.

In the example code, it is set to ‘000’ to use FRCOH0 and FRCOL0.

6.1.5 Range Comparator Inverted Range Select Setting

RCINVSEL is the range comparator inverted range select bit.

0: The comparison checks for ‘outside range’, that is, the over threshold and interrupt flags are set when the ADC result is above the upper threshold or below the lower threshold.

1: The comparison checks for ‘inside range’, that is, the interrupt flags are set when the ADC result is below or equal to the upper threshold and above or equal to the lower threshold.

In this example code, it is set to ‘0’.

6.1.6 Range Comparator Interrupt Enable Setting

RCIRQE0 is the range comparator interrupt enable bit.

0: Range comparator interrupt disable

1: Range comparator interrupt enable

In this example code, it is set to ‘0’ because the example does not need interruption.

6.2 Analog-to-Digital Conversion for Range Compare

See Code 12 for the example code of analog-to-digital conversion for range compare.

Code 12. Example Code of Analog-to-Digital Conversion for Range Compare

```
/********** Endless main loop **********/
 for(;;)
 {
     /********** Software Trigger **********/
     ADC12B0_CHCTRL0_SWTRG = 1;
     /********** Wait A/D Convert Completion **********/
     while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)
     {
         ClearWatchdog();
     }
     /********** Clear Conversion Done Flag **********/
     ADC12B0_CDONEIRQC0_CDONEIRQC0 = 1;
     if(ADC12B0_RCIRQ0_RCIRQ0 == 1)        <-- Range Comparator Interrupt Flag
     {
         /********** Outside range **********/
         /********** Clear Range Compare Flag **********/
         ADC12B0_RCIRQC0_RCIRQC0 = 1;
         /********** Get ADC Value **********/
         Volume = ADC12B0_CD0;
     }
     /********** Clear hardware watchdog **********/
     ClearWatchdog();
 }
```

6.2.1 Code Description

After analog-to-digital conversion, the range comparator Interrupt flag is checked.

If the value of the flag equals ‘1’ (this means the ADC value is out of range), the flag is cleared and the AD value is stored to the variable named “Volume.”

7 Pulse Detection Procedure

The result of the comparison from the range comparator can be filtered with the pulse detection function.

For every logical channel, the pulse detection function has a pair of reload registers to store the initial value for the positive and negative down counters (ADC12Bn_PCCTRL0 to 63.PCTPRL[7:0] and ADC12Bn_PCCTRL0 to 63.PCTNRL[4:0]). The positive and negative counters decrement on positive and negative events obtained from the result of the comparison done by the range comparator.

This function is usually used to avoid misdetections of abnormal voltage caused by noise and so on when monitoring voltage.

For parameters described in the Range Comparator Settings, an analog-to-digital conversion result greater than 2500 or less than 1500 is a positive event; one greater than or equal to 1500 and less than or equal to 2500 is a negative event.

This example application stores the analog-to-digital conversion result into the RAM in the case positive event has occurred five times in a row, and after that, stores the value constantly as shown in Figure 8.

The continuous number of positive event is interrupted when negative event occurs twice in a row, that is, the count of positive event continues even if negative event has occurred just one time as shown in Figure 9.

The physical setting of this application is the same as shown in Figure 1.

Ensure that you have configured the settings described in the Port Settings and ADC Global Settings sections.

Figure 8. Behavior of Pulse Detection application 1

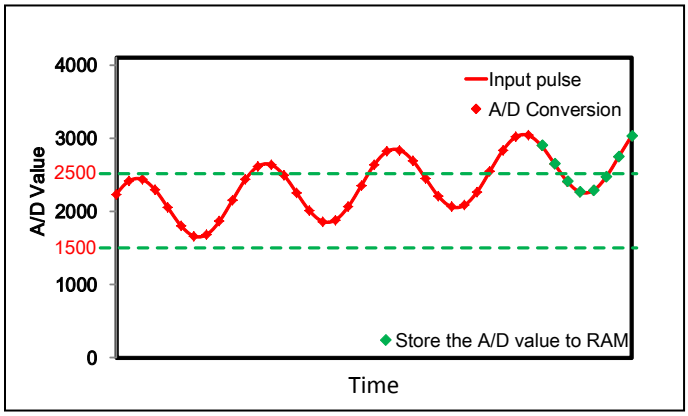
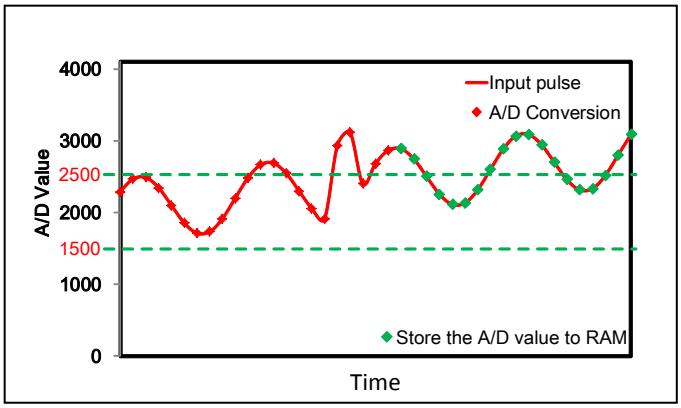


Figure 9. Behavior of Pulse Detection Application 2

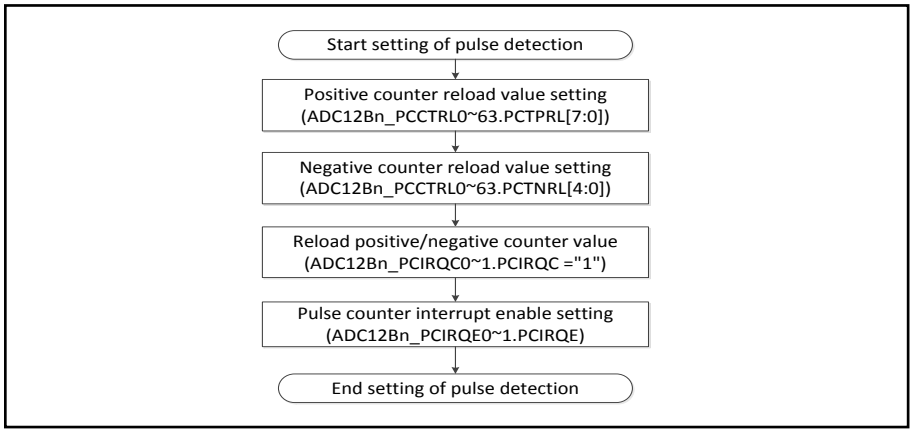


Use the information in the following sections and the code example to implement this application.

7.1 Pulse Detection Settings

Figure 10 shows the flow chart of pulse detection setting.

Figure 10. Pulse Detection Setting Flow Chart



Code 13 shows the example code of the pulse detection logical channel settings.

This code should also be implemented after Range Comparator Settings.

Code 13. Example Code of Pulse Detection Settings

```
/********** A/D Converter Pulse Ch0 Settings **********/
ADC12B0_PCCTRL0_PCTPRL = 5;     <-- Positive counter reload value setting
ADC12B0_PCCTRL0_PCTNRL = 2;     <-- Negative counter reload value setting
ADC12B0_PCIRQC0_PCIRQC0 = 1;    <-- Reload positive/negative counter value
ADC12B0_PCIRQE0_PCIRQE0 = 0;    <-- Pulse counter interrupt enable setting
```

7.1.1 Positive Counter Reload Value Setting

The ADC12B0_PCCTRL0_PCTPRL register holds the reload value of the positive counter register used for counting positive events of the range comparator. In this example, it is set to ‘5’.

7.1.2 Negative Counter Reload Value Setting

These register bits hold the reload value of the negative counter PCTNCT used for counting negative events of the range comparator. In this example, it is set to ‘2’.

7.1.3 Reload Positive/Negative Counter Value

ADC12B0_PCIRQC0_PCIRQC0 must be set to ‘1’ one time in the settings procedure, since positive / negative counter values are not loaded in the initial status.

7.1.4 Pulse Counter Interrupt Enable Setting

Disable pulse counter interrupt enable.

7.2 Analog-to-Digital Conversion and Output Pulse Detection Results

Code 14 gives the example code of the analog-to-digital conversion and output pulse detection results.

Code 14. Example Code of Analog-to-Digital Conversion and Pulse Detection Results

```
/********** Endless main loop **********/
for (;;)
{
    /********** Software Trigger **********/
    ADC12B0_CHCTRL0_SWTRG = 1;
    /********** Wait A/D Convert Completion **********/
    while(ADC12B0_STAT_BUSY)
    {
        ClearWatchdog();
    }
    /********** Clear Conversion Done Flag **********/
    ADC12B0_CDONEIRQC0_CDONEIRQC0 = 1;
    if(ADC12B0_PCIRQ0_PCIRQ0 == 1)      <-- Pulse Counter Interrupt Flag
    {
        /********** Get AD Value **********/
        Volume = ADC12B0_CD0;
    }
    /********** Clear hardware watchdog **********/
    ClearWatchdog();
}
```

7.2.1 Code Description

The pulse counter interrupt flag is checked after analog-to-digital conversion.

If the value of the flag equals ‘1’ (this means instances when the AD value is out of range have occurred five times in a row), the AD value is stored into the variable named ‘Volume’.

8 Diagnosis Function

This section shows flowcharts and example codes that explain how to use the diagnosis function on the ADC.

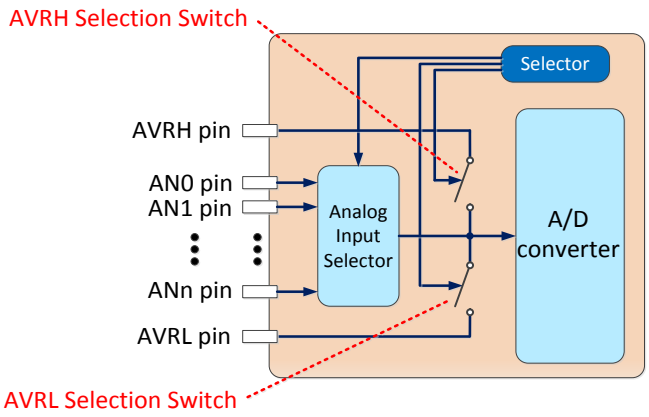
It is possible to switch voltage that is input into the ADC to reference voltages AVRH and AVRL, as shown in Figure 11.

AVRH is the upper-limit reference voltage. The AD value is 0xFFF (= 4095).

AVRL is the lower-limit reference voltage. The AD value is 0x000 (= 0).

These are the functions for ADC diagnosis and ADC calibration.

Figure 11. AVRH and AVRL Switch in ADC Peripheral



8.1 Check Maximum Value of Zero Transition Voltage and Minimum Value of Full-Scale Transition Voltage

To conduct ADC diagnosis, first see the *S6J3200 Series, Datasheet* to check the maximum value of the zero transition voltage (VZT) and the minimum value of the full-scale transition voltage (VFST), as shown in Table 4.

According to the datasheet, the maximum value of VZT is AVRL + 12.5 LSB. The AD value is 12.5 (0 + 12.5).

Therefore, when AVRL is input to the ADC, the value must be 12 or lower. (The AD value ≤ 12 = 0x00C.)

The minimum value of VFST is AVRH – 13.5 LSB. The AD value is 4081.5 (4095 – 13.5).

Therefore, when AVRH is input to the ADC, the value must be 4082 or more. (The AD value ≥ 4082 = 0xFF2.)

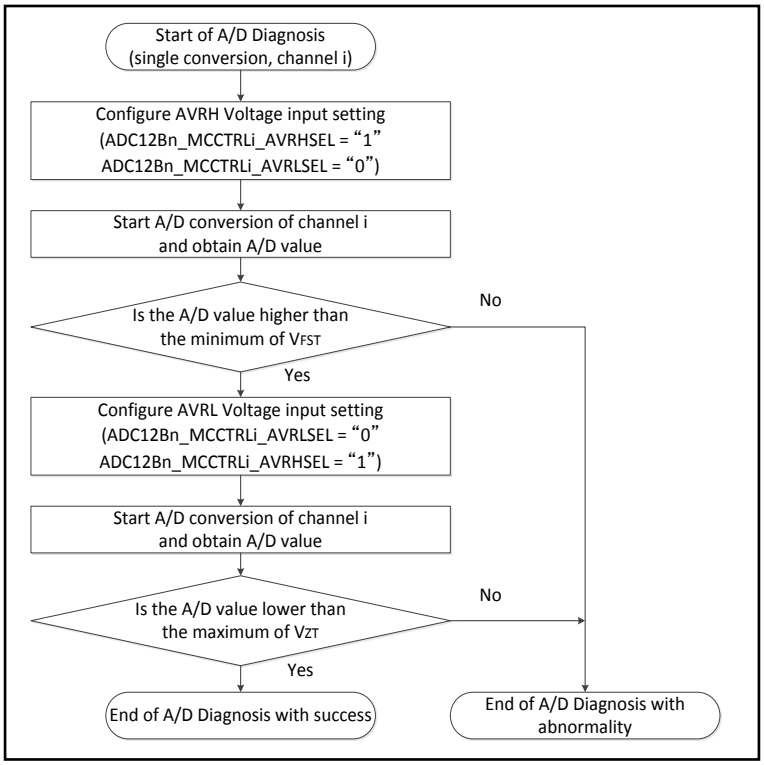
Table 4. Max VZT and Min VFST from Device ADC Datasheet

Parameter	Symbol	Pin Name	Value			Unit	Remarks
			Min	Typ	Max		
Resolution	-	-	-	-	12	bit	
Total Error	-	-	-	-	±12	LSB	
Integral Non linearity	-	-	-	-	±4.0	LSB	
Differential Non linearity	-	-	-	-	±1.9	LSB	
Zero transition voltage	V_{ZT}	AN0 to AN49	AVRL -11.5LSB	-	AVRL +12.5LSB	V	
Full-scale transition voltage	V_{FST}	AN0 to AN49	AVRH -13.5LSB	-	AVRH +10.5LSB	V	
Sampling time	t_{SMP}	-	0.3	-	-	µs	
Compare time	t_{CMP}	-	0.8	-	28	µs	
A/D conversion time	t_{CNV}	-	1.1	-	-	µs	
A/D trigger input time		ADTRG	$4t_{CLK\_LCP1A}$	-	-	ns	$4t_{CLK\_LCP1A} \geq$ 100ns
			100				$4t_{CLK\_LCP1A} <$ 100ns
Resumption time	-	-	-	-	1	us	-
Analog port input current	I_{AIN}	AN0 to AN17	-1.0	-	1.0	µA	$V_{AVSS} \leq V_{AIN} \leq V_{AVCC}$
		AN18 to AN25	-2.0	-	2.0	µA	
		AN26 to AN49	-3.0	-	3.0	µA	
Analog input voltage	V_{AIN}	AN0 to AN49	AVSS	-	AVRH	V	
Reference voltage	AVRH	AVRH5	4.5	-	5.5	V	$AV_{CC} \geq$ AVRH
	AVRL	AVRL5/AVSS	-	0.0	-	V	
Power supply current	I_A	AVCC	-	500	900	µA	
	I_{AH}		-	1.0	100	µA	
	I_R	AVRH	-	1.0	2.0	mA	
	I_{RH}		-	-	5.0	µA	
Variation between channels	-	AN0 to AN49	-	-	4.0	LSB	

8.2 Diagnosis Procedure

Figure 12 shows the diagnosis flow chart.

Figure 12. ADC Diagnosis Flow Chart



The example code of ADC diagnosis is given in Code 15.

Code 15. Example Code of ADC Diagnosis

```
#define AD_DIAGNOSIS_SUCCESS 0
#define AD_DIAGNOSIS_FAILURE 1
unsigned char AD_Diagnosis(void)
{
    unsigned short ADValue;
    /********** Switch analog input to AVRH **********/
    ADC12B0_MCCTRL0_AVRHSEL = 1;
    ADC12B0_MCCTRL0_AVRLSEL = 0;
    /********** Software Trigger **********/
    ADC12B0_CHCTRL0_SWTRG = 1;
    /********** Wait A/D Convert Completion **********/
    while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)
    {
        ClearWatchdog();
    }
    /********** Get AD Value **********/
    ADValue = ADC12B0_CD0;
    /********** Confirm the AD value is higher than VFSTmin **********/
```

```
    if(ADValue < 0xFF2)
    {
        /********** finish AD diagnosis with abnormality **********/
        ADC12B0_MCCTRL0_AVRHSEL = 0;
        ADC12B0_MCCTRL0_AVRLSEL = 0;
        return AD_DIAGNOSIS_FAILURE;
    }
    /********** Switch analog input to AVRL **********/
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    ADC12B0_MCCTRL0_AVRLSEL = 1;
    /********** Software Trigger **********/
    ADC12B0_CHCTRL0_SWTRG = 1;
    /********** Wait A/D Convert Completion **********/
    while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)
    {
        ClearWatchdog();
    }
    /********** Get AD Value **********/
    ADValue = ADC12B0_CD0;
    /********** Confirm the AD value is higher than VZTmax **********/
    if(ADValue > 0x00C)
    {
        /********** finish AD diagnosis with abnormality **********/
        ADC12B0_MCCTRL0_AVRHSEL = 0;
        ADC12B0_MCCTRL0_AVRLSEL = 0;
        return AD_DIAGNOSIS_FAILURE;
    }
    /********** finish AD diagnosis with success **********/
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    ADC12B0_MCCTRL0_AVRLSEL = 0;
    return AD_DIAGNOSIS_SUCCESS;
}
```

8.2.1 Set AD Input to AVRH

ADC12Bn_MCCTRLi_AVRHSEL is the analog-to-digital reference voltage AVRH selection bit.

0: AVRH voltage is not selected for analog-to-digital conversion.

1: AVRH voltage is selected for analog-to-digital conversion.

ADC12Bn_MCCTRLi_AVRLSEL is the analog-to-digital reference voltage AVRL selection bit

0: AVRL voltage is not selected for analog-to-digital conversion.

1: AVRL voltage is selected for analog-to-digital conversion.

If both the AVRHSEL and AVRLSEL bits are set to '1', AVRLSEL has a higher priority and the AVRL voltage is converted.

If any AVRHSEL/ARVRLSEL bits are set to '1', conversion of a regular analog input is not possible.

So set ADC12Bn_MCCTRLi_AVRHSEL to '1' and ADC12Bn_MCCTRLi_AVRHSEL to '0' to input AVRH into the ADC.

8.2.2 Start ADC and Obtain the Value

In this example, a software trigger is used.

8.2.3 Check Whether the Value Is Out of the Maximum Limit

The maximum value is specified in the datasheet (Table 4).

If the value is out of limit, finish the diagnosis with abnormality and return diagnosis fail error.

8.2.4 Set Analog-to-Digital Input to AVRHL

Set ADC12Bn_MCCTRLi_AVRHSEL to "0" and ADC12Bn_MCCTRLi_AVRHSEL to "1" to input AVRL into the ADC.

8.2.5 Check Whether the Value Is Out of the Minimum Limit

The minimum value is specified in the datasheet (Table 4).

If the value is out of limit, finish the diagnosis with abnormality and return diagnosis fail error.

9 Calibration Function

It is possible to adjust the offset and gain of the ADC using the ADC12Bn_OCV and ADC12Bn_GCV registers.

This section describes the effects of these registers and how to process the calibration of the ADC.

9.1 Offset Adjustment Direction

The ADC has an offset adjustment function to compensate for offset error.

The ADC12Bn_OVC register can adjust the offset.

It is possible to select code from +127 to –128 in a decimal number for ADC12Bn_OVC.

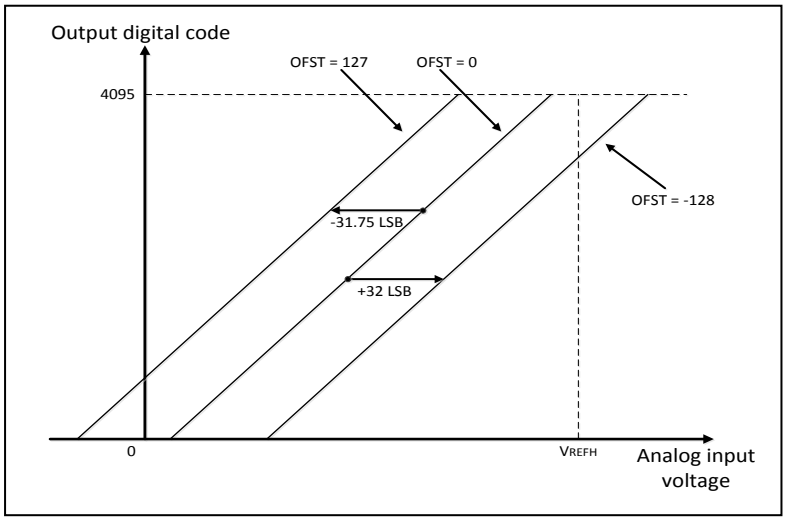
The offset adjustment step is a quarter of 1LSB.

The equation follows (OFST = the value of ADC12Bn_OVC).

$$\text{Output Digital Code} = \max\left(0, \min\left(4095, \text{floor}\left(\frac{\text{VIN}}{\text{VREFH}} \times 4096 + \frac{\text{OFST}}{4}\right)\right)\right)$$

The relationship between OFST and the offset shift direction is shown in Figure 13.

Figure 13. Relationship Between OFST and Offset Shift Direction



9.2 Gain Adjustment Direction

The ADC has a gain adjustment function to compensate for gain error.

The ADC12Bn_GVC register can adjust the gain.

It is possible to select code from +15 to –15 in a decimal number for ADC12Bn_GVC.

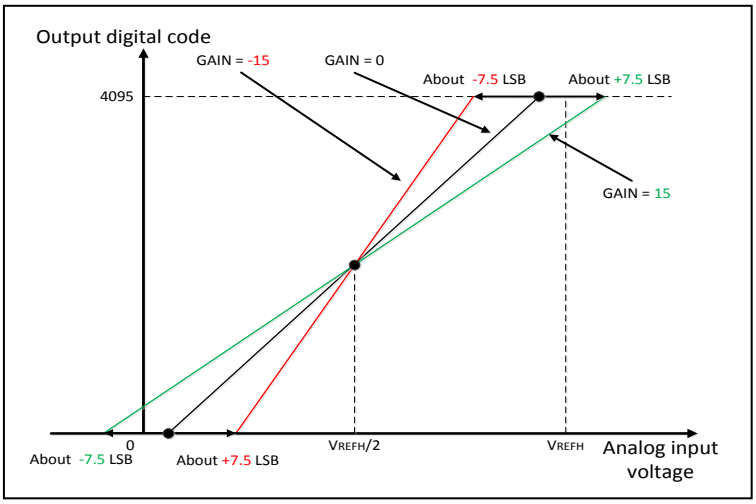
The gain adjustment step is about 1LSB.

The equation follows (GAIN= the value of ADC12Bn_GVC).

$$\text{Output Digial Code} = \max\left(0, \min\left(4095, \text{floor}\left(\frac{4096-\text{GAIN}}{4096} \times \left(VIN - \frac{\text{VREFH}}{2}\right) + 2048\right)\right)\right)$$

The relationship between the sign bit and the gain shift direction is shown in Figure 14.

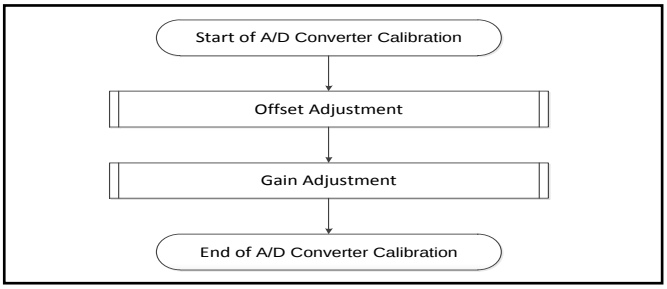
Figure 14. Relationship Between Sign Bit and Gain Shift Direction



9.3 Calibration Procedure

Figure 15 shows the flow chart of ADC calibration. First adjust the offset, and next adjust the gain.

Figure 15. ADC Offset and Gain Adjustment Flow Chart



The example code of ADC calibration is given in Code 16.

Code 16. Example Code of ADC Calibration

```
#define ADC_CALIBRATION_SUCCESS 0
#define ADC_CALIBRATION_FAULURE 1
#define ADC_OFFSET_SUCCESS 0
#define ADC_OFFSET_FAILURE 1
#define ADC_GAIN_SUCCESS 0
#define ADC_GAIN_FAILURE 1
unsigned char ADC_Calibration(void)
{
    /********** Conduct Offset Adjustment **********/
    if(ADC_Offset_Adjustment() != ADC_OFFSET_SUCCESS)
    {
        /********** Finish Calibration with abnormality **********/
        return ADC_CALIBRATION_FAULURE;
    }
    /********** Conduct Gain Adjustment **********/
    if(ADC_Gain_Adjustment() != ADC_GAIN_SUCCESS)
    {
        /********** Finish Calibration with abnormality **********/
        return ADC_CALIBRATION_FAULURE;
    }
    /********** Finish Calibration with success **********/
    return ADC_CALIBRATION_SUCCESS;
}
```

Offset adjustment

Finish calibration with abnormality

Gain adjustment

Finish calibration with abnormality

Finish calibration with success

9.3.1 Process ADC Offset Adjustment and Check Termination with Success

See the Offset Adjustment Procedure section for more information.

The function returns ‘ADC_OFFSET_SUCCESS’ if the offset adjustment finishes with success. If so, proceed to ADC gain adjustment.

If not, terminate the ADC calibration and report error.

9.3.2 Process ADC Gain Adjustment and Check Termination with Success

See the Offset Adjustment Procedure section for more information.

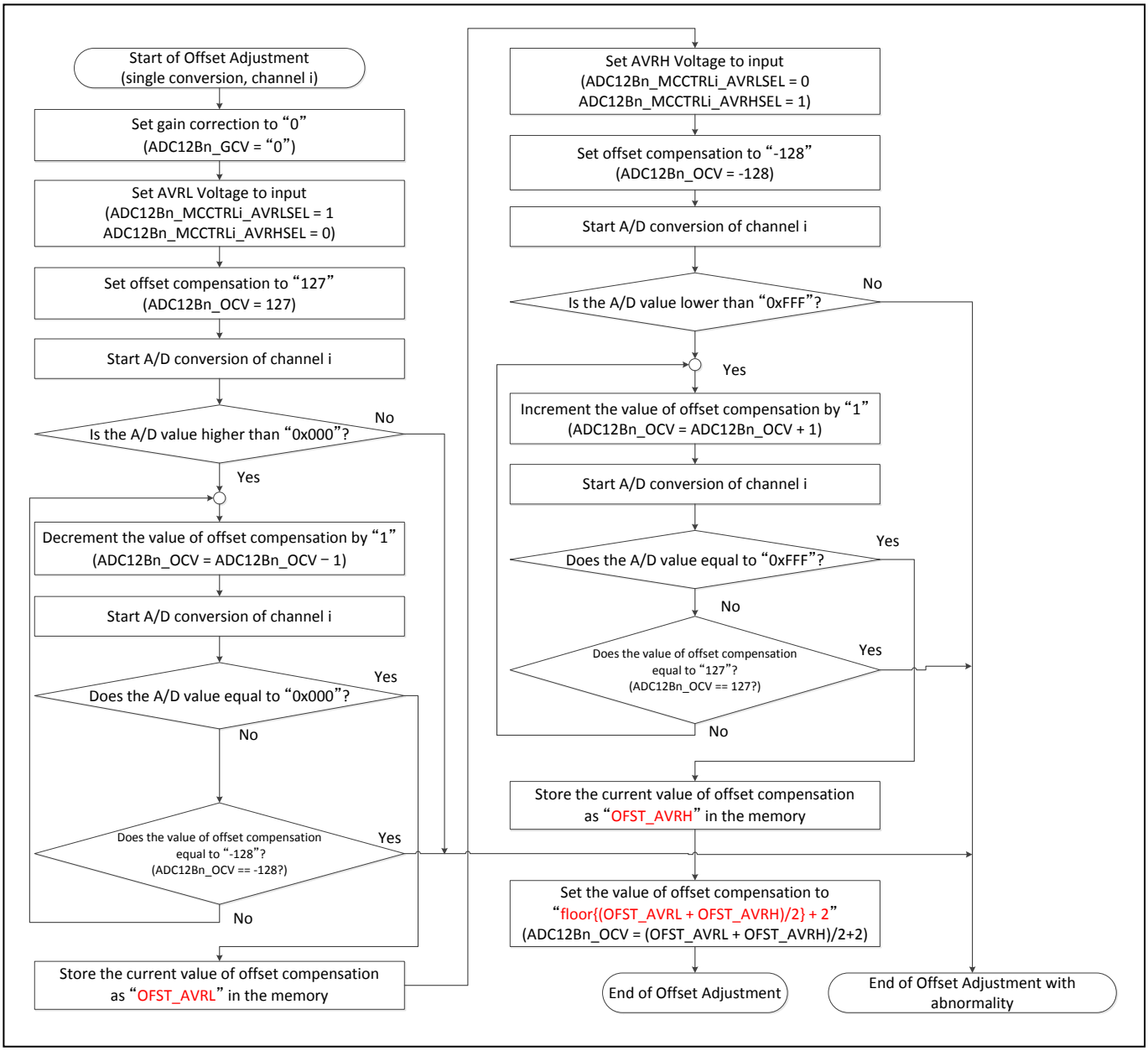
The function returns 'ADC_GAIN_SUCCESS' if the gain adjustment finishes with success.

If not, terminate the ADC calibration with abnormality and error.

9.4 Offset Adjustment Procedure

Figure 16 shows the flowchart of offset adjustment.

Figure 16. Offset Adjustment Procedure Flow Chart



The example code of offset adjustment is given in Code 17.

Code 17. Example Code of Offset Adjustment

```
#define OFFSET_REGISTER_VALUE_MAX (127)
#define OFFSET_REGISTER_VALUE_MIN (-128)
unsigned char ADC_Offset_Adjustment(void)
{
    signed short loop;
    unsigned short ADValue;
    signed char OFST_AVRL;
    signed char OFST_AVRH;

    /********** Set Offset Correction to "0" **********/
    ADC12B0_GCV = 0;
    /********** Switch analog input to AVRL **********/
    ADC12B0_MCCTRL0_AVRLSEL = 1;
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    for(loop=OFFSET_REGISTER_VALUE_MAX;loop>=OFFSET_REGISTER_VALUE_MIN;loop--)
    {
        ADC12B0_OCV = loop;
        /********** Software Trigger **********/
        ADC12B0_CHCTRL0_SWTRG = 1;
        /********** Wait A/D Convert Completion **********/
        while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)
        {
            ClearWatchdog();
        }
        /********** Get AD Value **********/
        ADValue = ADC12B0_CD0;
        if(loop == OFFSET_REGISTER_VALUE_MAX && ADValue == 0x000)
        {
            /********** Finish Offset Adjustment with abnormality **********/
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            return ADC_OFFSET_FAILURE;
        }
        else if(ADValue == 0x000)
        {
            OFST_AVRL = loop;
            break;
        }
        if(loop == OFFSET_REGISTER_VALUE_MIN)
        {
            /********** Finish Offset Adjustment with abnormality **********/
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            return ADC_OFFSET_FAILURE;
        }
```

```
    }
    /********** Switch analog input to AVRL **********/
    ADC12B0_MCCTRL0_AVRLSEL = 0;
    ADC12B0_MCCTRL0_AVRHSEL = 1;
    for(loop=OFFSET_REGISTER_VALUE_MIN;loop<=OFFSET_REGISTER_VALUE_MAX;loop++)
    {
        ADC12B0_OCV = loop;
        /********** Software Trigger **********/
        ADC12B0_CHCTRL0_SWTRG = 1;
        /********** Wait A/D Convert Completion **********/
        while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)
        {
            ClearWatchdog();
        }
        /********** Get AD Value **********/
        ADValue = ADC12B0_CD0;
        if(loop == OFFSET_REGISTER_VALUE_MIN && ADValue == 0xFFF)
        {
            /********** Finish Offset Adjustment with abnormality **********/
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            return ADC_OFFSET_FAILURE;
        }
        else if(ADValue == 0xFFF)
        {
            OFST_AVRH = loop;
            break;
        }
        if(loop == OFFSET_REGISTER_VALUE_MAX)
        {
            /********** Finish Offset Adjustment with abnormality **********/
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            return ADC_OFFSET_FAILURE;
        }
    }
    /********** Set to floor{(OFST_AVRL + OFST_AVRH) / 2} + 2 **********/
    ADC12B0_OCV = ((OFST_AVRL + OFST_AVRH) / 2) + 2;
    /********** Finish Offset Adjustment with success **********/
    ADC12B0_MCCTRL0_AVRLSEL = 0;
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    return ADC_OFFSET_SUCCESS;
}
```

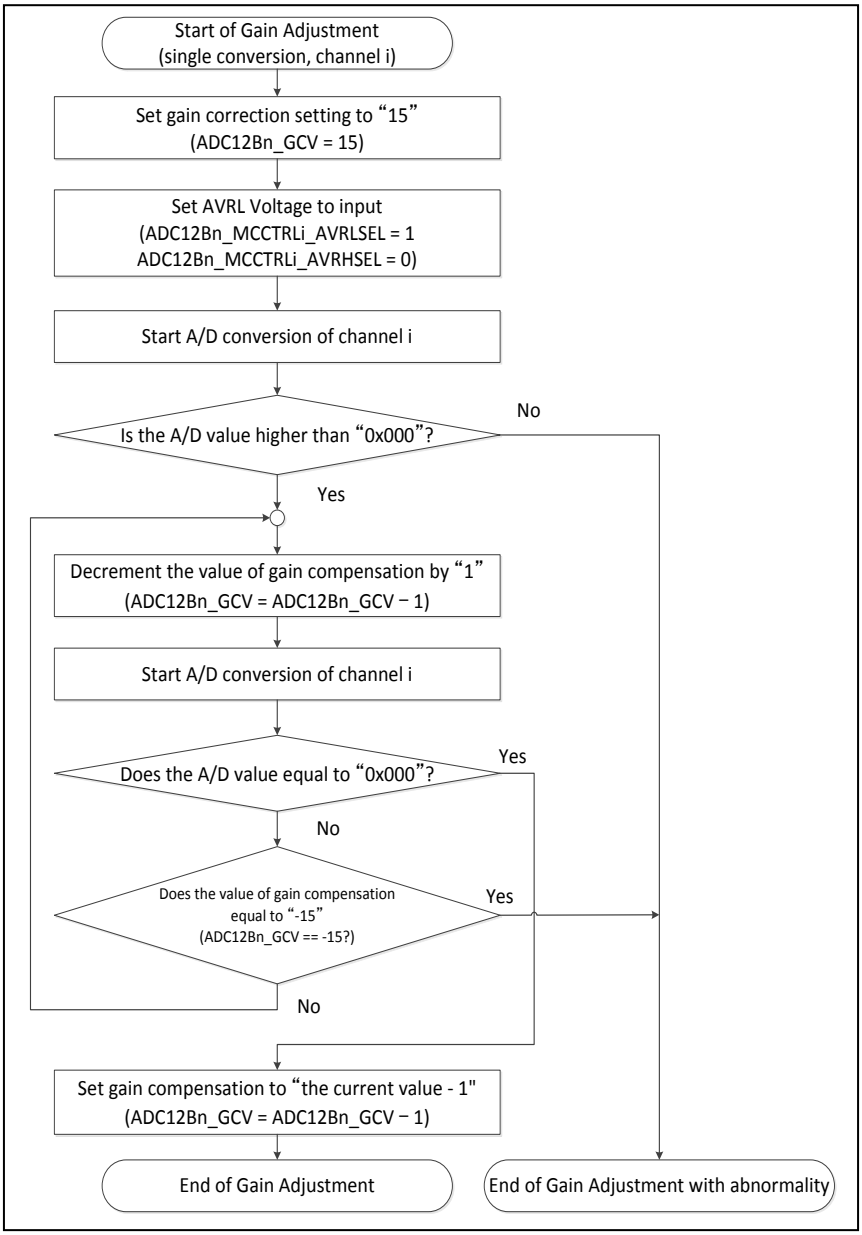
9.4.1 Code Description

This example code finds the appropriate offset value in accordance with the flow chart (Figure 16).

9.5 Gain Adjustment Procedure

Figure 17 shows the flow chart of gain adjustment.

Figure 17. Gain Adjustment Procedure Flow Chart



An example code of gain adjustment is given in Code 18.

Code 18. Example Code of Gain Adjustment

```
#define GAIN_REGISTER_VALUE_MAX (15)
#define GAIN_REGISTER_VALUE_MIN (-15)
unsigned char ADC_Gain_Adjustment(void)
{
    signed short loop;
    unsigned short ADValue;
    /********** Switch analog input to AVRL **********/
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    ADC12B0_MCCTRL0_AVRLSEL = 1;
    for(loop=GAIN_REGISTER_VALUE_MAX;loop>=GAIN_REGISTER_VALUE_MIN;loop--)
    {
        ADC12B0_GCV = loop;
        /********** Software Trigger **********/
        ADC12B0_CHCTRL0_SWTRG = 1;
        /********** Wait A/D Convert Completion **********/
        while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)
        {
            ClearWatchdog();
        }
        /********** Get AD Value **********/
        ADValue = ADC12B0_CD0;
        if(loop == GAIN_REGISTER_VALUE_MAX && ADValue == 0x000)
        {
            /********** Finish Gain Adjustment with abnormality **********/
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            return ADC_GAIN_FAILURE;
        }
        else if(ADValue == 0x000)
        {
            break;
        }
        if(loop == GAIN_REGISTER_VALUE_MIN)
        {
            /********** Finish Gain Adjustment with abnormality **********/
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            return ADC_GAIN_FAILURE;
        }
    }
    /********** Set Gain Compensation to "the current value - 1" **********/
    ADC12B0_GCV = loop - 1;
    /********** Finish Gain Adjustment with success **********/
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    ADC12B0_MCCTRL0_AVRLSEL = 0;
    return ADC_GAIN_SUCCESS;
}
```

9.5.1 Code Description

This example code finds the appropriate gain value in accordance with the flow chart (Figure 17).

10 Related Documents

Traveo family series datasheets and hardware manuals:

- S6J3200 Series Datasheet (Doc.No.002-05682)
- S6J3200 Series Hardware Manual (Doc.No.002-04852)
- S6J32E/F/G Series Datasheet (Doc.No.002-10689)
- S6J32E/F/G Series Hardware Manual (Doc.No.002-12500)
- Traveo Family Hardware Manual Platform Part for S6J3200 Series (Doc.No.002-04854)
- S6J3310/20/30/40 Series Datasheet (Doc.No.002-10635)
- S6J3350 Series Datasheet (Doc.No.002-10634)
- S6J3310/20/30/40/50 Series Hardware Manual (Doc.No.002-10185)
- Traveo Family Hardware Manual Platform Part for S6J3310/3320/3330/3340/3350 Series (Doc.No.002-07884)
- S6J3360/70 Series Datasheet (Doc.No.002-03359)
- S6J3360/70 Series Hardware Manual (Doc.No.002-18302)
- Traveo Family Hardware Manual Platform Part for S6J3360/3370 Series (Doc.No.002-07884)
- S6J3400 Series Datasheet (Doc.No.001-97829)
- S6J3400 Series Hardware Manual (Doc.No.002-09919)
- Traveo Family Hardware Manual Platform Part for S6J3400 Series (Doc.No.002-07884)
- S6J3510 Series Datasheet (Doc.No.002-18647)
- S6J3510 Series Hardware Manual (Doc.No.002-18642)
- Traveo Family Hardware Manual Platform Part for S6J3510 Series (Doc.No.002-07884)

Document History

Document Title: AN211319 - Using an ADC in Traveo™ Family S6J3200/ S6J3300/ S6J3350/ S6J3360/ S6J3370/ S6J3400/ S6J3510 Series

Document Number: 002-11319

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5230797	YOTS	05/13/2016	New application note.
*A	5392085	YOTS	09/15/2016	Added target products S6J3300/S6J3350/S6J3400 series Made compatible to the other application note. Updated template
*B	5554927	YOTS	12/15/2016	Bug fix and alignment of indents at some sample code.
*C	5697832	YOTS	05/09/2017	Added target products S6J3360/S6J3370/S6J3510 series. Minor modification in sample source code. Updated template

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2016-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.