EPSON
EXCEED YOUR VISION

# S1C17W22/S1C17W23 Thermohygrometer/ Salinometer (RFC) Application Notes

SEIKO EPSON CORPORATION

Rev. 1.1

## Evaluation board/kit and Development tool important notice

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

## NOTICE

## Summary

This document is intended to provide the reference material for measurements below using the R/F converter (RFC) built in S1C17W22 or S1C17W23.

1. Measure the temperatures and relative humidities by measuring the changes of resistances with the thermistor and humidity sensor.
2. Measure the salinity concentrations and temperatures of solutions by measuring the resistances between electrodes and the resistances of thermistor.

## Operating Environment

- S5U1C17W23T (SVT17W23: Software Evaluation Tool for S1C17W23, below)
  A dedicated cable is needed to connect this equipment to ICDmini.
- PC
  - Install the GNU17 (S5U1C17001C) development tool if not yet installed*
  - Install the ICDmini USB driver if not yet installed
- ICDmini (S5U1C17001H)
  A USB cable is needed to connect this equipment to PC.
- Programming packages (this one) for S1C17W22/S1C17W23
  - Programming package for evaluating count values from thermohygro sensor
    Also used for evaluating count values responding to resistances between electrodes as salinity concentrations
    (s1c17w22_w23_rfc_temp_hum_analy_gnu17vx)
  - Programming package for testing converted values from temperatures and humidities
    (s1c17w22_w23_rfc_therm_hygro_meter_gnu17vx)
  - Programming package for testing converted values from salinity concentrations
    (s1c17w22_w23_rfc_saltConc_meter_gnu17vx)

  Note: The behavior of this package is confirmed with GNU17 v2.4.0/v3.0.5.

# Table of Contents

# 1. Specification

This application notes describes the measurements by using the R/F converter (RFC) built in S1C17W22/S1C17W23.

1. Measure the temperatures and relative humidities by measuring the changes of resistances with the thermistor and humidity sensor.
2. Measure the salinity concentrations and temperatures of solutions by measuring the resistances between electrodes and the resistances of thermistor.

One of sample programs, s1c17w22_w23_rfc_temp_hum_analy_gnu17vx, evaluates the measured counts by changing the value of time base counter to change the resistances of thermistor or humidity sensor. The program was created mainly for evaluating the changes of sensor resistances by changing the temperature or humidity. This sample program is also available for evaluating count values responding to resistances between electrodes as salinity concentrations.

Another one, s1c17w22_w23_rfc_therm_hygro_meter_gnu17vx, repeatedly measures the resistances of thermohygro sensor at certain intervals of time with a 16bit timer (T16) and displays converted temperatures and humidities on LCD.

Similarly, another sample program, s1c17w22_w23_rfc_saltConc_meter_gnu17vx, repeatedly measures resistances between electrodes and resistances of thermistor at certain intervals of time with a 16bit timer (T16) and displays converted salinity concentrations corrected by using temperatures, converted temperature, and power supply voltages measured with SVD on LCD.

# 2. Description of Functions

RFC Ch.0 — Measures resistances by using the AC oscillation mode to handle the humidity sensor, AC-bias resistive sensor. Also, when measuring resistances between electrodes, measures resistances by using the AC oscillation mode to minimize the impact from the capacity because of electric double layer.

RFC Ch.1 — Measures resistances by using the DC oscillation mode to handle the thermistor, DC-bias resistive sensor.

T16 Ch.0 — This interval timer starts a measurement with RFC at certain intervals of time to get the resistances of thermohygro sensor or, the resistances between electrodes and the resistances of thermistor.

Here, describes the example of s1c17w22_w23_rfc_therm_hygro_meter_gnu17vx:

LCD24 — Displays the temperatures and humidities converted from resistances of thermohygro sensor on LCD.

COPRO2 — Used to multiply and divide values at high speed.

Operation mode — RFC Ch.0: The measurement counter values are gotten based on resistances of humidity sensor by using the interrupt by completion of the sensor A oscillation.
RFC ch.1: The measurement counter values are gotten based on resistances of thermistor by using the interrupt by completion of the sensor A oscillation.

System clock — Uses OSC1 (32.768kHz).

Interrupt — The RFC Ch.0 vector number and vector address are as follows:

RFC Ch.0 Vector number: 23 (0x17)
Vector address: 0x805c

This sample uses the following four kinds of interrupts.

- Interrupt by completion of reference oscillation
- Interrupt by completion of the sensor A oscillation
- Interrupt by overflow error of measurement counter
- Interrupt by overflow error of time base counter

The RFC Ch.1 vector number and vector address are as follows:

RFC Ch.1 Vector number: 24 (0x18)
Vector address: 0x8060

This sample uses the following four kinds of interrupts.

- Interrupt by completion of reference oscillation
- Interrupt by completion of the sensor A oscillation
- Interrupt by overflow error of measurement counter
- Interrupt by overflow error of time base counter

The T16 Ch.0 vector number and vector address are as follows:

T16 Ch.0 Vector number: 9 (0x09)
Vector address: 0x8024

This sample uses the following interrupt.

- Interrupt by underflow

Here, describes the example of s1c17w22_w23_rfc_saltConc_meter_gnu17vx:

SVD — Measures the power supply voltages for confirmation which may give an impact on resistances between electrodes.

LCD24 — Displays measured salinity concentrations, temperatures converted from resistances of thermistor, and power supply voltages measured with SVD on LCD.

| | |
|---|---|
| COPRO2 | Used to multiply and divide values at high speed. |
| Operation mode | RFC Ch.0: The measurement counter values are gotten based on resistances between electrodes by using the interrupt by completion of the sensor A oscillation.<br>RFC ch.1: The measurement counter values are gotten based on resistances of thermistor by using the interrupt by completion of the sensor A oscillation. |
| System clock | Uses OSC1 (32.768kHz). |
| Interrupt | The RFC Ch.0 vector number and vector address are as follows:<br>RFC Ch.0 Vector number: 23 (0x17)<br>Vector address: 0x805c<br>This sample uses the following four kinds of interrupts.<br>Interrupt by completion of reference oscillation<br>Interrupt by completion of the sensor A oscillation<br>Interrupt by overflow error of measurement counter<br>Interrupt by overflow error of time base counter<br><br>The RFC Ch.1 vector number and vector address are as follows:<br>RFC Ch.1 Vector number: 24 (0x18)<br>Vector address: 0x8060<br>This sample uses the following four kinds of interrupts.<br>Interrupt by completion of reference oscillation<br>Interrupt by completion of the sensor A oscillation<br>Interrupt by overflow error of measurement counter<br>Interrupt by overflow error of time base counter<br><br>The T16 Ch.0 vector number and vector address are as follows:<br>T16 Ch.0 Vector number: 9 (0x09)<br>Vector address: 0x8024<br>This sample uses the following interrupt.<br>Interrupt by underflow |

The Figure 1 shows the RFC configuration.

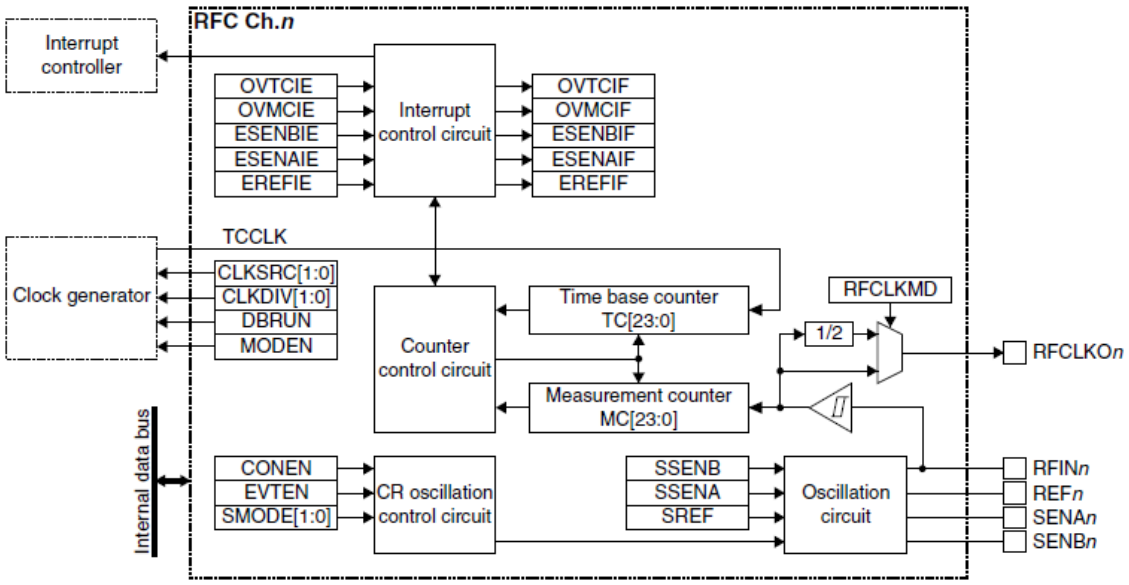



Figure 1 RFC configuration

# 3. Operating Principle

This chapter describes the principle of the R/F converter, temperature and humidity measurements, and salinity concentration measurement.

## 3.1 Measuring Temperature and Humidity Using R/F Converter

An A/D converter of CR oscillation type is built in the S1C17 family and is called an R/F converter. The sensor connected to the R/F converter measures the oscillation clock converted in the CR oscillation circuit by using the measurement counter. This count value is digital. As shown in Figure 3-1, initial value (0x000000-n) is set in the measurement counter. The counter calculates the time to overflow based on the oscillation of reference resistance (reference oscillation). Next, based on this time, the number of clocks oscillated (sensor oscillation) by resistive thermohygro sensor is counted until the time base counter underflows.

As for the thermistor, the temperatures are calculated by using these count values and the table prepared.

In the case of the humidity sensor, relative humidities are calculated by using measured temperatures, count values of humidity sensor, and the table prepared. That is, since relative humidity is dependent on temperature, the relative humidity should be corrected with the temperature.

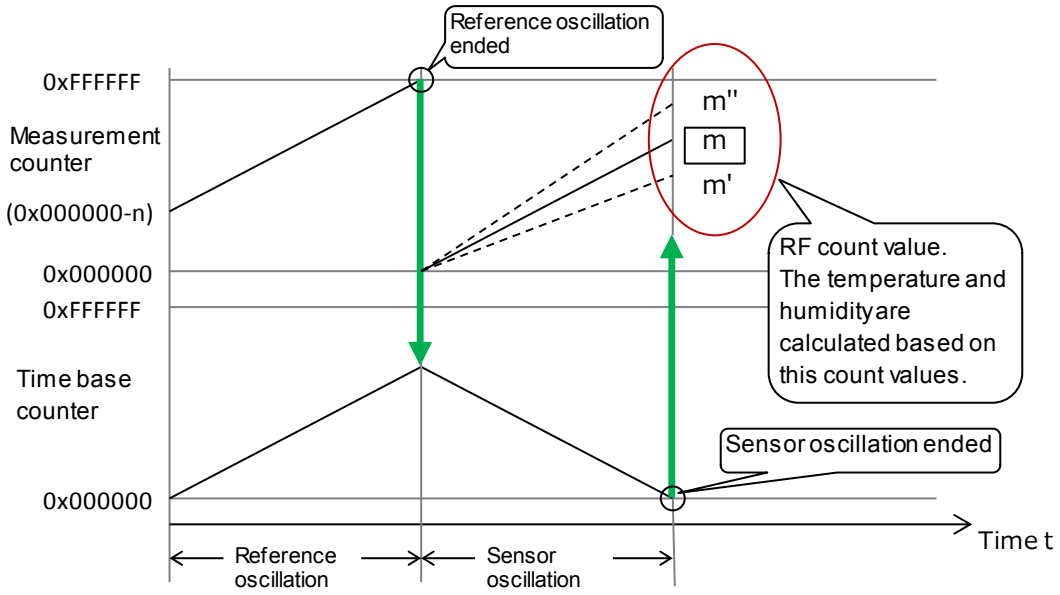



Figure 3-1 Behavior of R/F converter (reference oscillation and sensor oscillation)

The sample software measures temperatures in the DC oscillation mode of the R/F converter connected to a thermistor (103AT-2B manufactured by SEMITEC Corporation. Also, the count values are acquired in the AC oscillation mode of the R/F converter connected to resistive humidity sensor (CL-M52R, manufactured by Shinyei Technology). The relative humidities are determined by primary interpolating the humidity conversion table loaded in the sample software, measured temperatures, and count values of humidity sensor.

The sample software is based on the humidity/temperature measurement circuit shown in Figures 3-2, 3-3 and Table 3-1. Note that the part symbols in this circuit are assigned based on the symbols used in SVT17W23 (S5U1C17W23T).

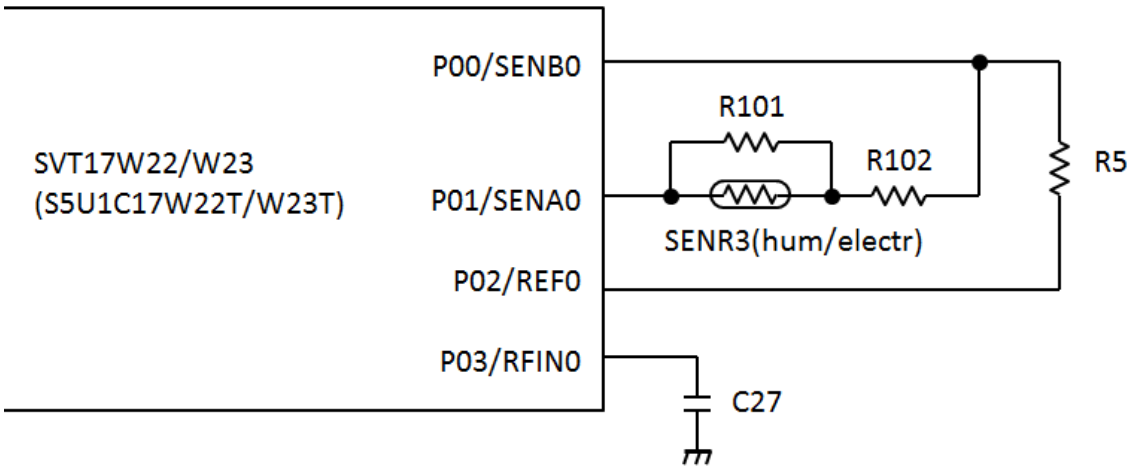



Figure 3-2 R/F converter Ch0 for measurement with humidity sensor/salinity concentration sensor in AC oscillation mode

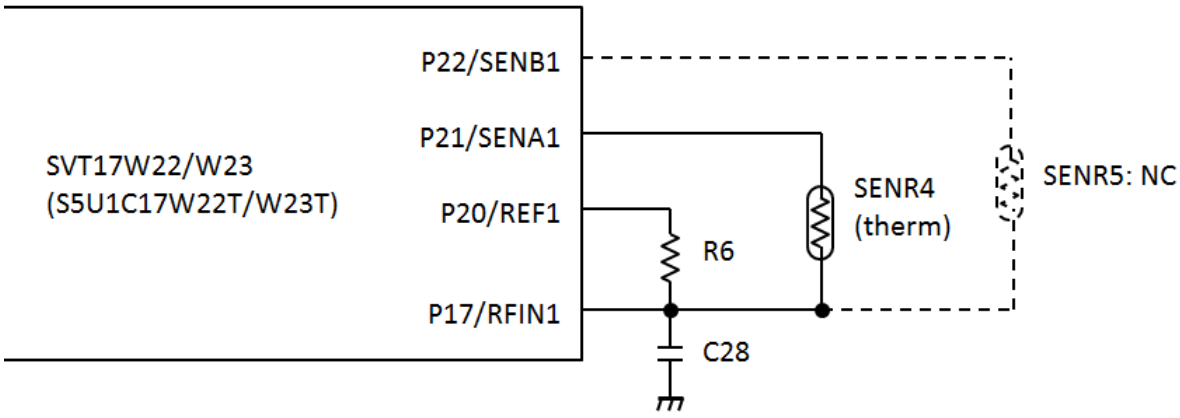



Figure 3-3 R/F converter Ch1 for measurement with thermistor in DC oscillation mode

Since the dynamic range of resistive humidity sensor is wide from several kΩ (high temperature/humidity) to tens MΩ (low temperature/humidity), in the range of low resistance, the resistance inside IC gives an impact. In the range of high resistance, the error increases as the count value decreases. In order to reduce these impacts, it's needed to add parallel resistance R101 and series resistance R102 so that adaptive range can be attained. When designing the software, create the humidity conversion table by taking care of these things.

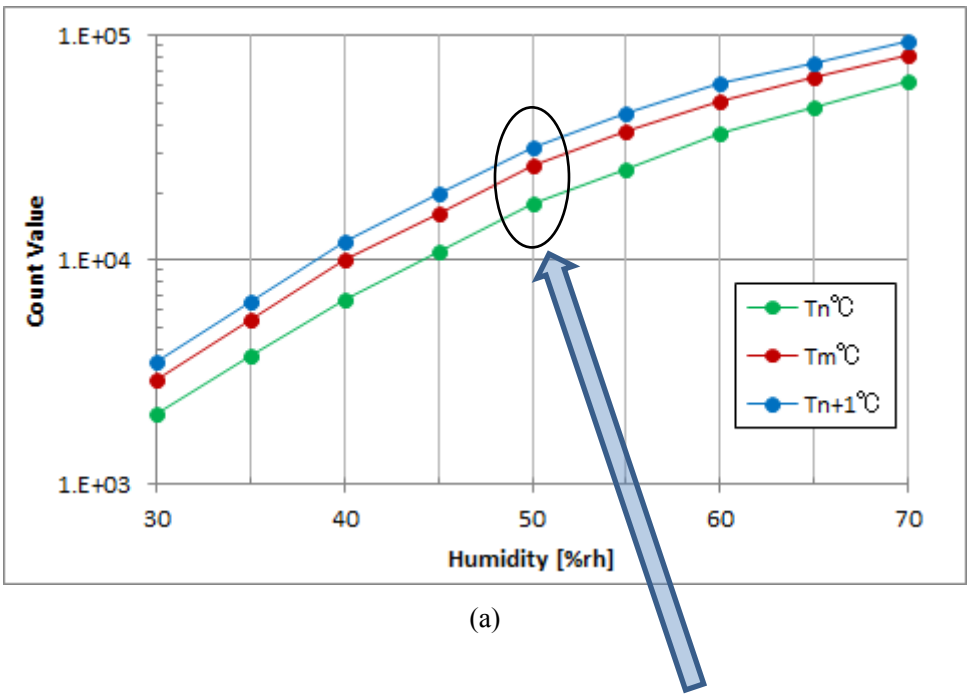
Also, when evaluating functions using SVT17W23 (S5U1C17W23T), in order to minimize the wire length, mount parts on the RF converter parts area (Ch0/Ch1) where part symbols are silk printed on the board and do not use extended interface connector J1. When the lead wire of parts may be long beyond necessity, take care of the possibility that the impact of mechanical vibration may make the measurement values uncertain.

Table 3-1 BOM of R/F converters in Figures 3-2 (humidity measurement) and 3-3

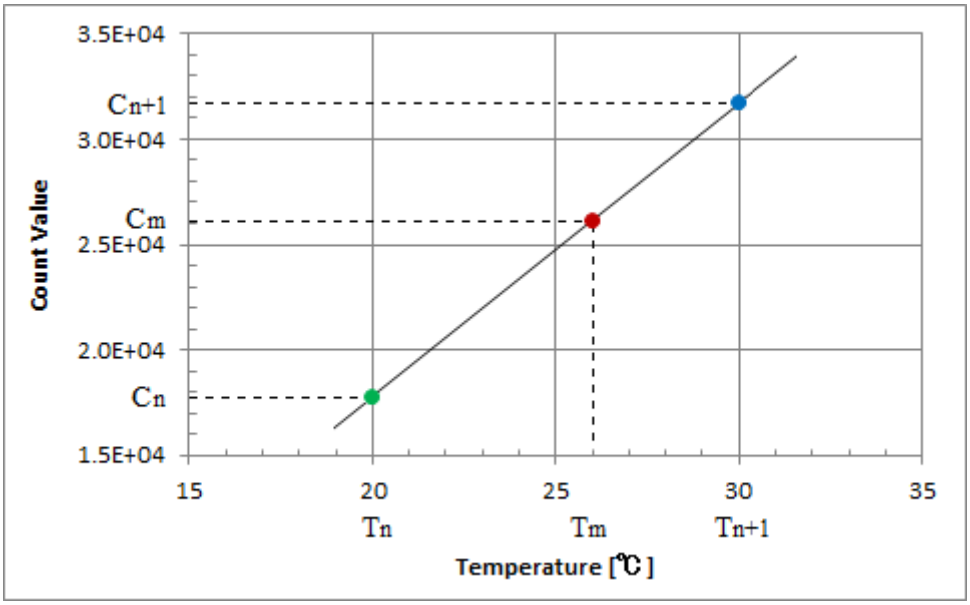
| Product Name | Symbol | Manufacturer | Part Number of Manufacturer | Specification |
|---|---|---|---|---|
| Humidity sensor | SENR3 | Shinyei Technology | CL-M52R | 31.0kΩ(typ.) at 25°C |
| Thermistor | SENR4 | SEMITEC Corporation | 103AT-2B | 10.18kΩ ±1% at 25°C |
| Capacitor | C27, C28 | muRata | GRM1552C1H102JA01 | 1000pF/50V 1005 CH |
| Resistance | R5 | ROHM | MCR01 Series / MCR006YRTx Series | 47kΩ 0603 F |
| | R6 | | | 10kΩ 0603 F |
| | R101 | | | 10MΩ 0603 F |
| | R102 | | | 4.3kΩ 0603 F |

The use of spec code C0G, CH with smaller temperature coefficient as ceramic capacitor in Table 3-1 (BOM) may be unlikely to cause a problem for measuring temperatures and humidities.

## 3.2 Principle of Calculating Humidity




(a)




(b)

Figure 3-4 Correcting count values responding to humidity with temperature

When converting resistances measured with thermistor into temperatures, the conversion is simply possible by proportionally calculating resistances based on the table of resistances measured at a regular temperature interval. However, during the conversion, take care of the fact that the resistances of humidity sensor may change according to the temperature as well as humidity.

Therefore, use temperatures measured with thermistor and then create the table data used for humidity conversion by primary interpolating two temperature tables adjacent to temperatures measured. For example, if Tm=26°C is given and the temperature intervals of tables is 10°C, two tables Cn and Cn+1 are used with count values: Tn=20°C and Tn+1=30°C. Based on these tables, the count value Cm corresponding to Tm=26°C is calculated. (Though the vertical axis of graph (a) is logarithmic scale in Figure 3-4, but when computing interpolation, the vertical axis is handled as linear scale, as shown in graph (b) in Figure 3-4. The reason is to avoid the large load of logarithmic calculation.)

$$Cm = Cn + (Tm - Tn) \cdot \frac{Cn+1 - Cn}{Tn+1 - Tn}$$

Cm: Count value calculated in primary interpolation

Cn: Count value for temperature Tn = Table value prepared beforehand

Cn+1: Count value for temperature Tn+1 = Table value prepared beforehand

Tm: Temperature measured with thermistor

Tn: Boundary temperature of temperature range (at 10°C intervals: 0°C, 10°C, ..., 50°C)

Tn+1: Boundary temperature of temperature range at Tn + 10°C

## 3.3 How to Measure Resistance of Humidity Sensor

Since the humidity sensor used in this application notes uses polymer membranes for detecting humidities, AC voltage should be applied for measuring resistances.

Strictly speaking, the way to measure resistances with RFC does not apply AC voltage, so the behavior of count values was researched by actually applying AC voltage from RFC while checking whether there is any problem with this RFC method. At the same time, to check how the frequency and waveform of AC voltage changes that are applied to the humidity sensor, we observed the output with oscilloscope by assigning terminal P40 to RFCLKO0.

The Figure 3-4 shows how the values (measured count) are dependent on the humidity when the value n ("n for MC", below) which is used in initial value (0x000000-n) for measurement counter is changed at 50°C ambient temperature. Also, the Figure 3-5 shows how the frequencies of RFCLKO0 are dependent on the humidity at the time of measurement above.

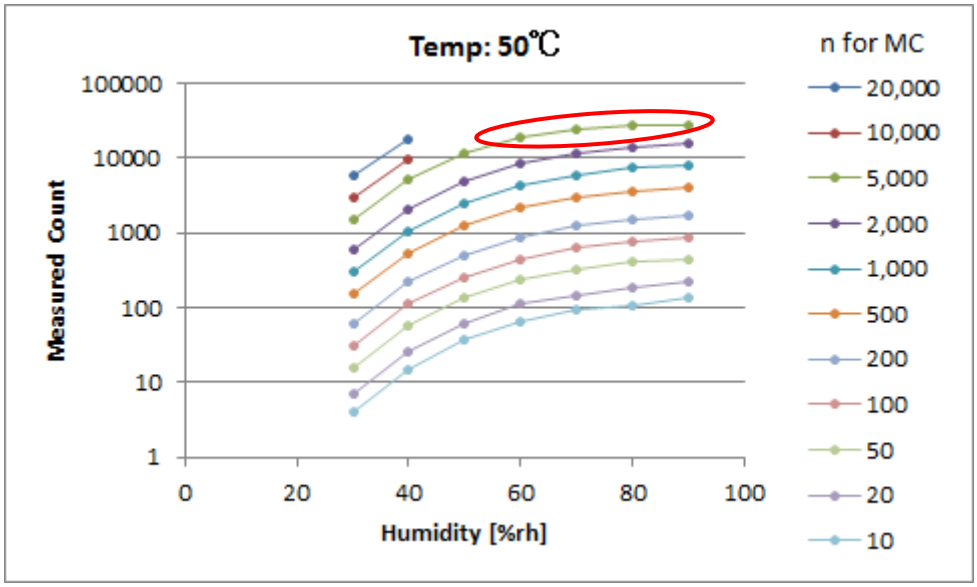



Figure 3-4 How measured counts are dependent on humidity in the humidity sensor

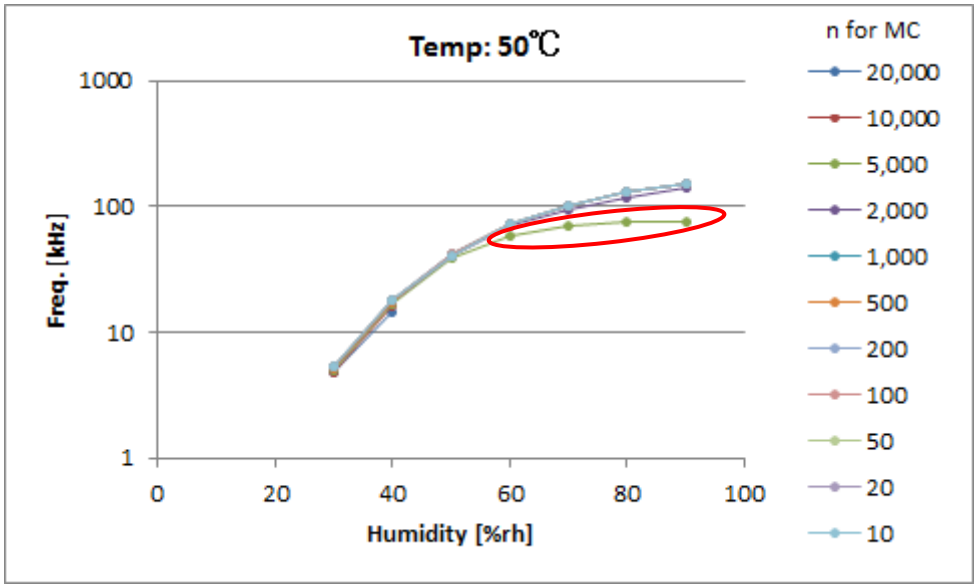



Figure 3-5 How frequencies of RFCLKO0 are dependent on humidity in the humidity sensor

The figure shows that the data in the area enclosed with a red circle in both graphs is different in trend from other data.

The data in the area enclosed in Figure 3-4 indicates the results for n=5000 and represents that the response to humidity is dropping and the slope of the curve is becoming lower. Also, the slope of the curve of the result for n=2000 below it also looks like becoming lower. These behaviors seem to happen when the measurement counter value (Measured count on the vertical axis. Not "n for MC" for reference oscillation) exceeds 10000.

Also, according to Figure 3-5, the change of the dependency of frequencies of RFCLKO0 on humidity is different from changes under other conditions and the frequency will not increase along with the increase of humidity. (Notes: It was already confirmed with the LCR meter that the change of frequency between 100Hz and 100kHz and the change of AC bias between 0.1V and 1.0V cause only a small change between 340 kΩ and 500kΩ to the resistance of humidity sensor under the circumstance of 25°C and 30%rh. Even if the AC frequency applied to the humidity sensor changes during measurement, the change does not give a big impact on resistance.)

The Figure 3-6 shows the AC voltage waveforms of RFCLKO0 applied to the humidity sensor when this phenomenon occurs. The waveforms are output for a half frequency clock by setting RFC0CTL.RFCLKMD register value to 1. The waveforms are monitored just before the oscillation of AC applied to the humidity sensor stops.

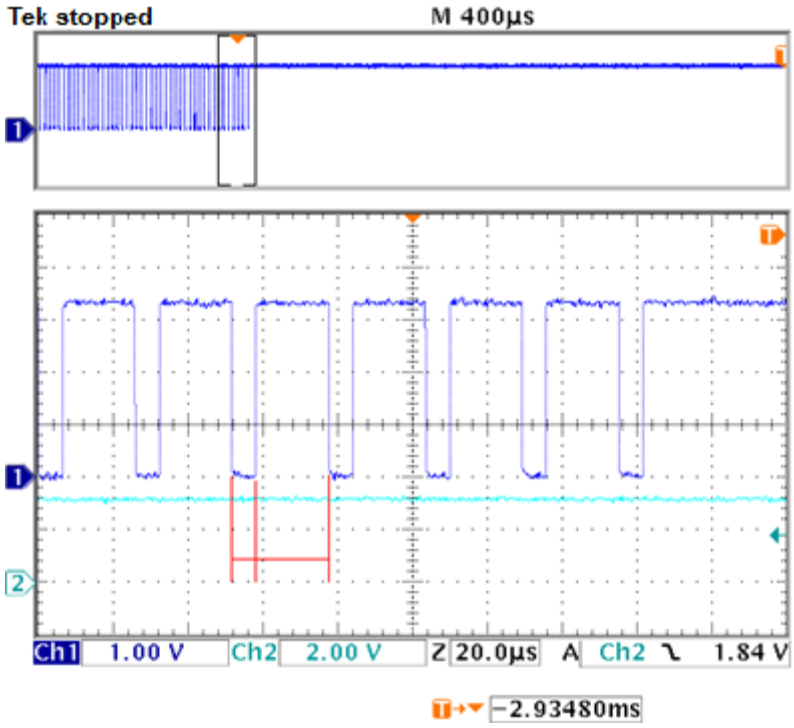



Figure 3-6 Monitoring of AC waveforms of RFCLKO0 applied to humidity sensor

The duty ratio is 50% for a while after the start of oscillation, but just before the oscillation stops, it is observed that the duty ratio deviates from 50% a lot. (If the value n increases further, the duty ratio deviates significantly.)

Thus, the application of AC voltage while the duty ratio deviates a lot from 50% does not satisfy the specification (AC waveform application) of the humidity sensor of polymer membrane type. As the measured count increases by increasing the value n, the count becomes more unstable. In particular, the measured count changes gradually during measurement so that the count will not converge on a certain value. Therefore, it's necessary to avoid such an operating condition and the necessary condition is: Measured count should not exceed 10000.

On the other hand, if measured count is too small, the accuracy may be degraded because of quantization error. The resistances of humidity sensor should be measured using RFC so that measured counts are within certain range. The practical values are described in "Appendix A. Example of creating thermohygro table."

## 3.4 Measuring Salinity Concentration Using R/F Converter

Basically, the way to use R/F oscillator is the same as the section describing measuring temperature and humidity. Especially, the way to measure temperatures of solution is totally the same except that the temperature range and intervals of the table used are different.

The salinometer in this application notes has been manufactured mainly by taking care of how much the measuring error can be reduced after researching the impart of solution temperature on measured values and performing the correction. So, the salinity concentrations are calculated by using measured temperatures, count values of salinity concentration sensor, and the table prepared in advance. The power supply voltage is also measured by using SVD and displayed which gives an impact on the salinity concentrations calculated.

The sample software gets count values by using the R/F converter in AC oscillation mode that is connected to electrodes as a salinity concentration sensor. The salinity concentrations are determined by primary interpolating the salinity concentrations conversion table loaded in the sample software, measured temperatures, and count values of salinity concentration sensor.

The sample software is based on the salinity concentration & temperature measurement circuit shown in Figures 3-2, 3-3 and Table 3-2. Note that the part symbols in this circuit are assigned based on the symbols used in SVT17W23 (S5U1C17W23T).

The electrodes used here as salinity concentration sensor may not satisfy the specification for electrical characteristics of the R/F converter ($R_{REF}$(min)=1kΩ) because their resistance will be too low under high temperature and high salinity concentration conditions. When the electrodes are not soaked in the solution and stay in the air, the resistance is infinitely great. Though the state can be detected under the condition RFC0INTF.OVMCIF bit=1 (measurement counter overflow error), it's necessary to check how to distinguish it from failures including disconnection. Therefore, similar to humidity sensor, parallel resistance R101 and series resistance R102 were added. When designing the software, create the salinity concentration conversion table by taking care of these things.

Also, when evaluating functions using SVT17W23 (S5U1C17W23T), in order to minimize the wire length, similar to thermohygrometer, mount parts on the RF converter parts area (Ch0/Ch1) where part symbols are silk printed on the board and do not use extended interface connector J1. When the lead wire of parts may be long beyond necessity, take care of the possibility that the impact of mechanical vibration may make the measurement values uncertain.

Table 3-2 BOM of R/F converters in Figures 3-2 (salinity concentration measurement) and 3-3

| Product name | Symbol | Manufacturer | Part Number of Manufacturer | Specification |
|---|---|---|---|---|
| Salinity concentration sensor | SENR3 | - | - | See APPENDIX B. |
| Thermistor | SENR4 | SEMITEC Corporation | 103AT-2B | 10.18kΩ ±1% at 25°C |
| Capacitor | C27, C28 | muRata | GRM1552C1H102JA01 | 1000pF/50V 1005 CH |
| Resistance | R5 | ROHM | MCR01 Series | 2.2kΩ 0603 F |
| | R6 | | | 10kΩ 0603 F |
| | R101 | | | 1MΩ 0603 F |
| | R102 | | | 1kΩ 0603 F |

The use of spec code C0G, CH with smaller temperature coefficient as ceramic capacitor in Table 3-2 (BOM) may be unlikely to cause a problem for measuring salinity concentrations.

## 3.5 Principle of Measuring Salinity Concentration

This application notes is supposed to measure the salinity concentrations of daily foods. Roughly speaking, the conductivity and salinity concentration are proportional to each other, so it's possible to use RFC as a simple salinometer when the accuracy is not required very much. The salinity concentration sensor uses a pair of electrodes and works as a salinometer by measuring the values corresponding to electric resistances between electrodes using the R/F converter.

However, since there is electric double layer on the interface between the solution and electrodes, an equivalent circuit is created between electrodes as shown in Figure 3-7. In Figure 3-8 Nyquist plot (Cole-Cole plot), minute AC voltage with changed frequency is applied to this equivalent circuit and the impedances (Z) are plotted on based on the current response characteristics.

The impact of electric double layer is removed by applying AC waveforms of considerably high frequency to electrodes so that only the electric resistances Rs of solution can be extracted that may change according to salinity concentrations.

Thus, the measurement of resistances with AC enables to measure resistances of solution without altering the composition of specimen by applying high DC voltage which may cause the electrolysis.

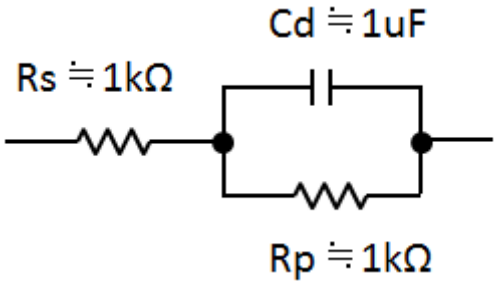



Figure 3-7 Equivalent circuit of electrodes in solution

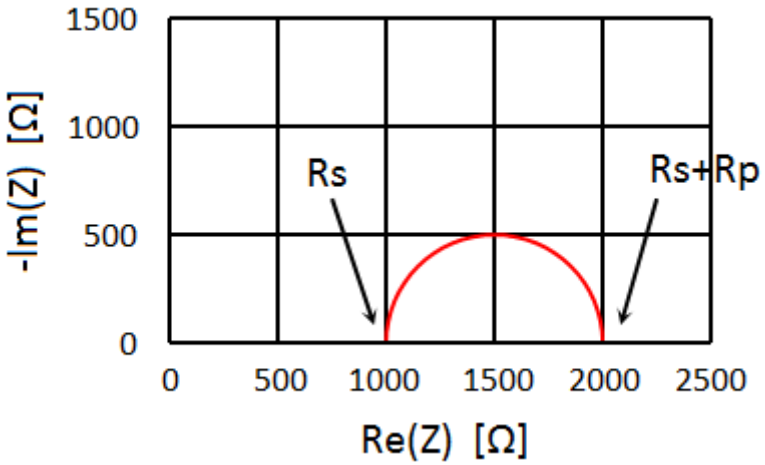



Figure 3-8 Nyquist plot (Cole-Cole plot) of equivalent circuit of electrodes in solution

However, similar to humidity sensor, take care of the fact that the resistances between electrodes soaked in solution may change according to the temperature as well as salinity concentrations.

Therefore, use temperatures of solution measured with thermistor and then create the table data used for salinity concentration conversion by primary interpolating two temperature tables adjacent to temperatures measured. The calculation is the same as that of the humidity calculation, so refer to "3.2. Principle of Calculating Humidity."

# 4. Software Description

## 4.1 About s1c17w22_w23_rfc_temp_hum_analy_gnu17vx

This section describes the software, s1c17w22_w23_rfc_temp_hum_analy_gnu17vx, which has been created mainly to be used for evaluating the changes of sensor resistances based on the change of temperature and humidity and evaluates the measured counts based on the change of resistances on thermistor or humidity sensor by changing the time base counter values.

### 4.1.1 File Configuration

| Filename | Function |
|---|---|
| src/boot.c | Startup module file |
| src/clg.c | CLG driver file |
| src/init.c | Initialization function file |
| src/main.c | Main function file |
| src/rfc.c | RFC driver file |
| src/t16_ch0.c | T16 Ch.0 driver file |
| src/crt0.h | Boot processing definition file (for GNU17 Version 3) |
| src/clg.h | CLG driver header definition file |
| src/init.h | Initialization function header definition file |
| src/rfc.h | RFC driver header definition file |
| src/t16_ch0.h | T16 Ch.0 driver header definition file |
| inc/reg | Folder for storing peripheral device register definition files |
| inc/c17w22_reg.h<br>inc/c17w23_reg.h | Peripheral device header definition files |

### 4.1.2 Module Description

This section mainly describes non-generic function names and their functions.

File name: main.c

| Function Name | Function |
|---|---|
| measRfc | Updates time base counter value and then starts the RFC conversion. |
| intT16Ch0 | T16 Ch.0 interrupt function. Calls the function measRfc above. |

### 4.1.3 Structure

This section describes structures defined in the sample programs.

Structure name: TEMP_HUM_ANALY

| Variable Name | Type | Function |
|---|---|---|
| stat0 | unsigned short | Status of RFC Ch.0. |
| stat1 | unsigned short | Status of RFC Ch.1. |
| measCountIndex0 | unsigned short | Index of initial value (0x000000-n) for measurement counter for putting it in RFC Ch.0. |
| measCountIndex1 | unsigned short | Index of initial value (0x000000-n) for measurement counter for putting it in RFC Ch.1. |
| dataPtr0 | unsigned short | Pointer which indicates the location number (#) in said structure's sensorCount0*[#] for putting the measured value of RFC Ch.0. |
| dataPtr1 | unsigned short | Pointer which indicates the location number (#) in said structure's sensorCount1*[#] for putting the measured value of RFC Ch.1. |
| sensorCount0L | unsigned short | Low-order 16bit of said structure's sensorCount0. |
| sensorCount0H | unsigned short | High-order 8bit of said structure's sensorCount0. |
| sensorCount1L | unsigned short | Low-order 16bit of said structure's sensorCount1. |
| sensorCount1H | unsigned short | High-order 8bit of said structure's sensorCount1. |
| measCount0 | unsigned long | Initial value (0x000000-n) for measurement counter for putting it in RFC Ch.0. |
| measCount1 | unsigned long | Initial value (0x000000-n) for measurement counter for putting it in RFC Ch.1. |
| sensorCount0 | unsigned long | Sensor count measured using RFC Ch.0. |
| sensorCount1 | unsigned long | Sensor count measured using RFC Ch.1. |

### 4.1.4 Global Variable

This section describes global variables used in the sample programs.

File name: src/rfc.c

| Variable Name | Type | Function |
|---|---|---|
| tha | TEMP_HUM_ANALY | Structure variable for holding setting values and measured values for RFC Ch.0 and Ch.1. |

### 4.1.5 Operating Procedures

The sample software includes two projects for GNU17 Version 2 (hereinafter referred to as GNU17v2) and GNU17 Version 3 (hereinafter referred to as GNU17v3), and two header file sets for S1C17W22 and S1C17W23.

Before the sample software for GNU17v2 or GNU17v3 can be used, configure the target model by following the procedure shown below.

(1) Copy the header files for the target model to be used to the inc folder of the sample software.

Example: If the target model is S1C17W23, copy the c17w23_reg.h file and the reg folder to the inc subfolder in the s1c17w22_w23_rfc_temp_hum_analy_gnu17vx folder of the sample software.

(2) Select [Import...] from the [File] menu to start the Import Wizard, then select

[General] > [Existing Project to Workspace] (GNU17v2) or

[General] > [Existing Projects into Workspace] (GNU17v3).

(3) Select the project folder that contains the sample program:

s1c17w22_w23_rfc_temp_hum_analy_gnu17v2 folder (GNU17v2) or

s1c17w22_w23_rfc_temp_hum_analy_gnu17v3 folder (GNU17v3).

(4) Select [Copy projects into workspace] and then click the [Finish] button to exit the Import Wizard.

(5) Change the target CPU.

(GNU17v2)

1. Select the imported project in the [C/C++ project] view and select [Properties] from the [Project] menu.
2. Select [GNU17 General] from the property list in the [Properties] dialog box that appears.
3. Select the target CPU from the [Target CPU Device] drop-down list.
4. Click the [Apply] button.

(GNU17v3)

1. Select the imported project in the [Project Explorer] view and select [Properties] from the [Project] menu.
2. Select [GNU17 Setting] from the property list in the [Properties] dialog box that appears.
3. Select the target CPU from the [Target CPU] drop-down list.
4. Click the [OK] button to close the dialog box. Then, go to Step (7).

(6) Set the debugger's startup options. (GNU17v2 only)

1. Select [GNU17 GDB Commands] from the property list.
2. Click the [Create commands from template] button to display the [Create a simple startup command] dialog box.
3. Select "ICD Mini" from the [Debugger:] drop-down list and select [Execute flash ROM writing].
4. Click the [Overwrite] button to close the dialog box. Close the [Properties] dialog box as well.

(7) Edit the init.h header file that exists in the src folder of the project to change the target model defined.

Example: When the target is S1C17W23

```
//#define MCUSEL_C17W22       /// S1C17W22
#define MCUSEL_C17W23         /// S1C17W23
```

(8) Build the project.

Use IDE to build the s1c17w22_w23_rfc_temp_hum_analy_gnu17vx project.

(9) Connect the equipment and turn on the power.

1. Connect SVT17W23, ICDmini, USB cable, and PC with each other.
2. Reset SVT17W23 and ICDmini.

(10) Run the sample software.

Use IDE to run the s1c17w22_w23_rfc_temp_hum_analy_gnu17vx project.

### 4.1.6 Understanding Thermohygro Sensor Characteristics with Interactive Operation

This sample program is configured to measure the sensor's counter value by changing the value n dynamically which is used in initial value (0x000000-n) for measurement counter.

Change the tha.measCountIndex0 and tha.measCountIndex1 values from 0 (n=10) to 18 (n=10M) so that the value n which is used in initial value (0x000000-n) for measurement counter can be either 10, 20, 50, 100, ..., 5M, or 10M. That is, the value n is configured so that the value (0x000000-n) is needless to be set in each case.

According to the default T16_CH0_INTERVAL_TIME value, this sample program measures the resistance of thermohygro sensor every five seconds and stores the result in memory. Then, this program stores 16 (=RFC_DATA_SET_NUM) pieces of measured values in memory, so all storage areas will be updated with latest measurement data after running this program for 75seconds (5sec x (16 - 1)). The 16 sets of measurement values can be collected at a time by stopping the program when it went by 75sec after starting the program.

In practice, each time collecting measurement values, change tha.measCountIndex0 and tha.measCountIndex1 to desired values. So, the sensor count data can be collected after changing the value n of initial value (0x000000-n) for measurement counter.

Note that, to change the value of structure tha, just enter tha in Watch Expressions Window, push the [Add Watch] button, and change values of corresponding variables.

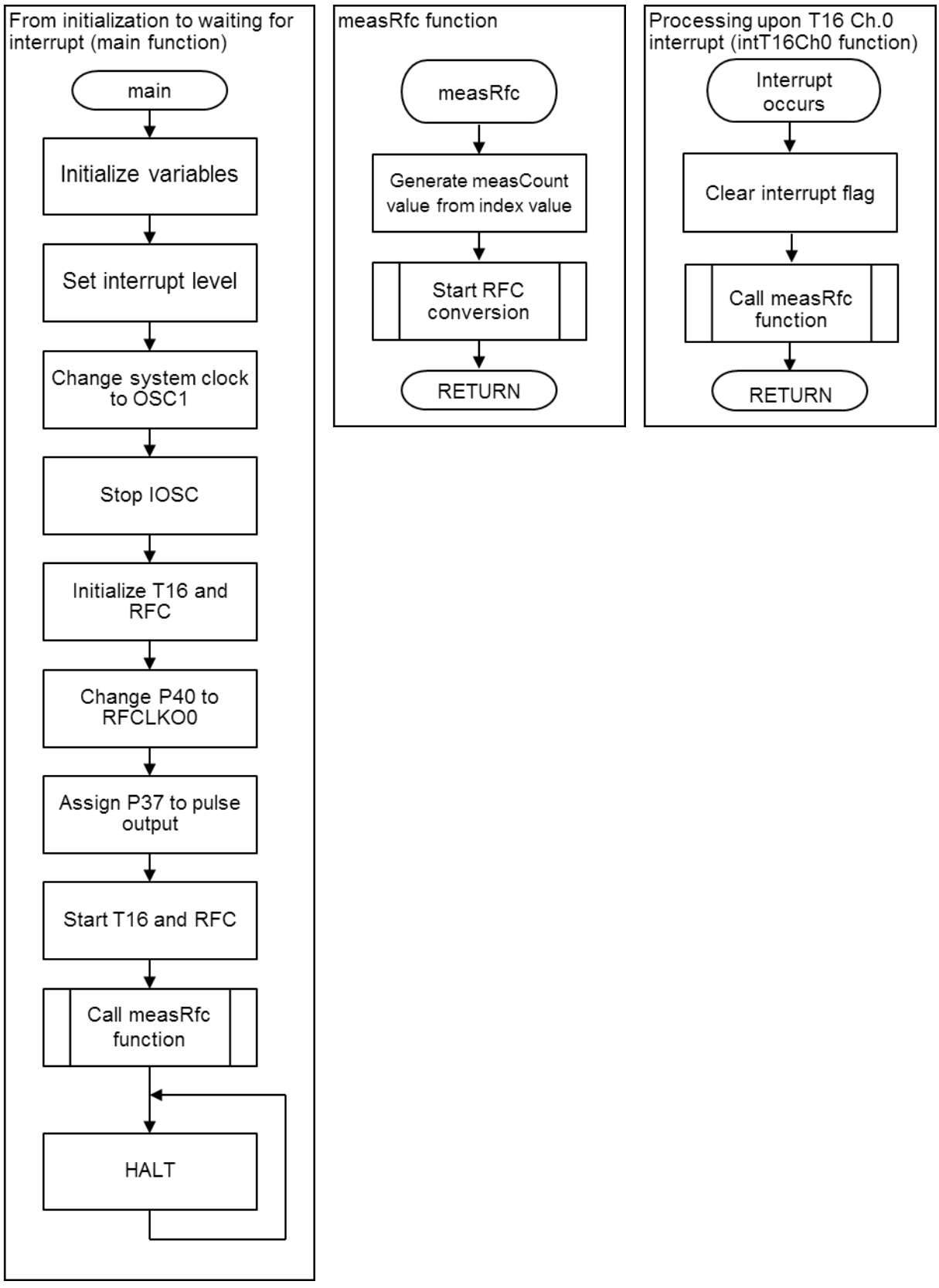
### 4.1.7 Sample Program Operation Overview

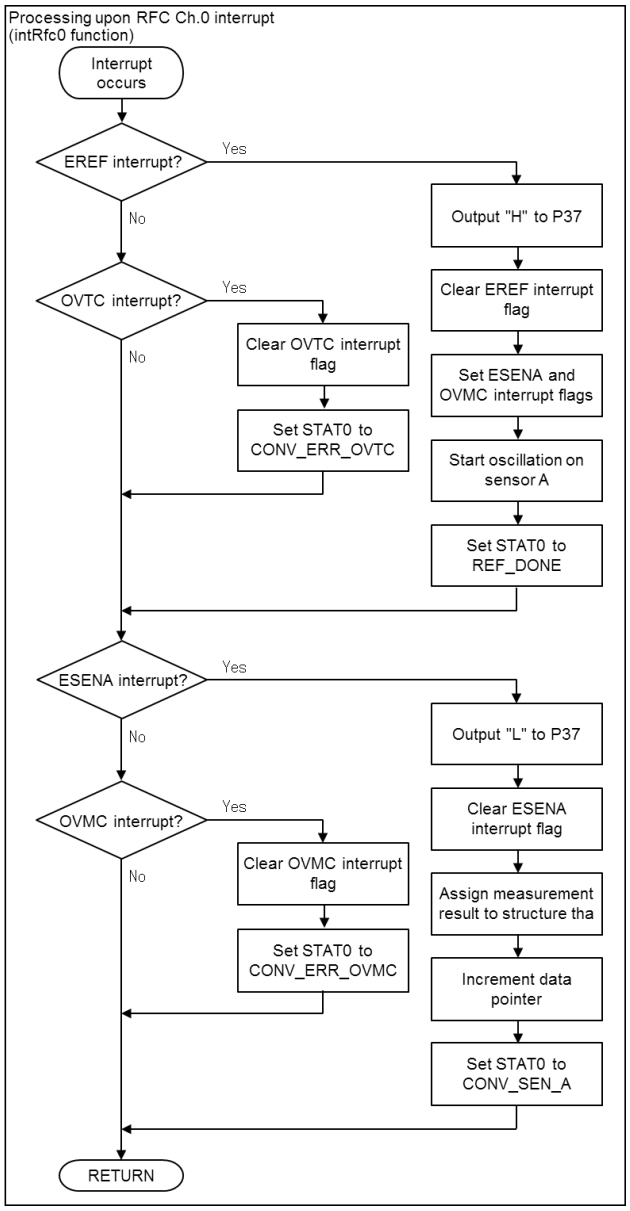
(1) Initialize parameters and interrupt level.
(2) Switch the system clock from IOSC to OSC1 (32.768kHz) and stop IOSC.
(3) Initialize T16 Ch.0, RFC Ch.0, and RFC Ch.1, change over P40 to the RFCLKO0 signal output mode, set P37 to conversion period monitor pulse output mode of RFC Ch.0.
(4) Start to count T16 Ch.0, launch RFC Ch.0 and RFC Ch.1, and call the measurement startup function.
(5) Repeat the HALT loop to wait for the interrupt of T16 Ch.0, RFC Ch.0, and RFC Ch.1.
(6) Call the measurement startup function each time the program is interrupted by T16 Ch.0.

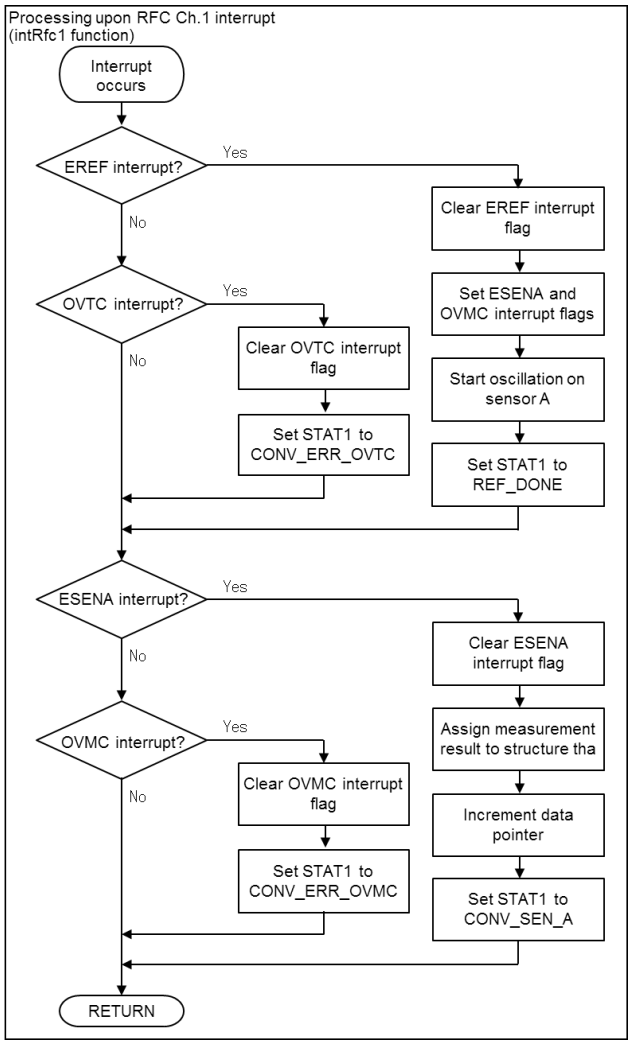
Here, describes the behavior of the measurement startup function for RFC Ch.0 and RFC Ch.1 in the step 4 above.

(1) The initial values (0x000000-n) for measurement counter are set to tha.measCount0 and tha.measCount1 according to the values of tha.measCountIndex0 and tha.measCountIndex1.
(2) Start the measurement using RFC Ch.0 and RFC Ch.1.
(3) Each time RFC Ch.0 or RFC Ch.1 interrupts, the status is updated based on the interrupt factor analysis and the sensor count data is collected. When RFC Ch.0 interrupts, the pulse for conversion period monitor is output to P37.

The figure below shows the flow chart:




Processing upon RFC Ch.0 interrupt
(intRfc0 function)
Interrupt occurs
EREF interrupt?
Yes
No
Output "H" to P37
Clear EREF interrupt flag
Set ESENA and OVMC interrupt flags
Start oscillation on sensor A
Set STAT0 to REF_DONE
OVTC interrupt?
Yes
No
Clear OVTC interrupt flag
Set STAT0 to CONV_ERR_OVTC
ESENA interrupt?
Yes
No
Output "L" to P37
Clear ESENA interrupt flag
Assign measurement result to structure tha
Increment data pointer
Set STAT0 to CONV_SEN_A
OVMC interrupt?
Yes
No
Clear OVMC interrupt flag
Set STAT0 to CONV_ERR_OVMC
RETURN


Processing upon RFC Ch.1 interrupt
(intRfc1 function)
Interrupt occurs
EREF interrupt?
Yes
No
Clear EREF interrupt flag
Set ESENA and OVMC interrupt flags
Start oscillation on sensor A
Set STAT1 to REF_DONE
OVTC interrupt?
Yes
No
Clear OVTC interrupt flag
Set STAT1 to CONV_ERR_OVTC
ESENA interrupt?
Yes
No
Clear ESENA interrupt flag
Assign measurement result to structure tha
Increment data pointer
Set STAT1 to CONV_SEN_A
OVMC interrupt?
Yes
No
Clear OVMC interrupt flag
Set STAT1 to CONV_ERR_OVMC
RETURN

## 4.2 About s1c17w22_w23_rfc_therm_hygro_meter_gnu17vx

This section describes the software, s1c17w22_w23_rfc_therm_hygro_meter_gnu17vx, which converts RFC measured counts into temperature and humidity and displays them on LCD. The RFC measured counts change according to the change of resistances of thermohygro sensor.

### 4.2.1 File Configuration

| Filename | Function |
|---|---|
| src/boot.c | Startup module file |
| src/clg.c | CLG driver file |
| src/display.c | LCD display function file |
| src/init.c | Initialization function file |
| src/main.c | Main function file |
| src/rfc_tbl.c | Conversion table of RFC temperature and humidity, function file |
| src/rfc.c | RFC driver file |
| src/t16_ch0.c | T16 Ch.0 driver file |
| src/crt0.h | Boot processing definition file (for GNU17 Version 3) |
| src/clg.h | CLG driver header definition file |
| src/display.h | LCD display function header definition file |
| src/init.h | Initialization function header definition file |
| src/lcd_font.h | LCD display font definition file |
| src/rfc_tbl.h | Conversion header definition file of RFC temperature and humidity |
| src/rfc.h | RFC driver header definition file |
| src/t16_ch0.h | T16 Ch.0 driver header definition file |
| inc/reg | Folder for storing peripheral device register definition files |
| inc/c17w22_reg.h<br>inc/c17w23_reg.h | Peripheral device header definition files |

### 4.2.2 Module Description

This section mainly describes non-generic function names, their functions, array variable names, and their details.

File name: display.c

| Function Name | Function |
|---|---|
| selectLcdFont | Returns a pointer to specified LCD font array. |
| displayChar | Displays a character at specified row and position on LCD. |
| displayStr | Displays a string at specified row on LCD. |
| clearLcdLine | Clears the display at specified row on LCD. |

File name: main.c

| Function Name | Function |
|---|---|
| initLcd | Initializes LCD mounted on SVT17W23. |
| measRfc | Updates time base counter value of RFC Ch.0 for measuring humidity and then starts the conversion of RFC Ch.0 and RFC Ch.1. |
| intT16Ch0 | T16 Ch.0 interrupt function. Calls the function measRfc above. |
| val2Str3 | Converts the value of unsigned short type into up to three digits of numeric value and substitutes the value into specified position of string. |
| dispVal | Displays the temperature and humidity on LCD. When the value is out of scope, displays accordingly. |

File name: rfc_tbl.c

| Array Variable Name | Description |
|---|---|
| thermSensCount | Count value of temperature sensor in the range from -20°C to 50°C at 1°C intervals. Measured value of RF count when n is 20000 (=THERM_MEAS_COUNT) which is used in initial value (0x000000-n) for measurement counter for RFC Ch.1. |
| hygroCoef | Count value of the humidity sensor in the range from 0°C to 50°C at 10°C intervals and in the range from 10%rh to 90%rh at 5%rh step. Measured value of RF count when n is 16384 (=HYGRO_COEF_MULT) which is used in initial values (0x000000-n) for measurement counter for RFC Ch.0. |

File name: rfc_tbl.c

| Function Name | Function |
|---|---|
| initRfcTbl | Initializes array elements of global structure variable tht. |
| getTemp | Calculates the temperature based on the sensor count measured using RFC Ch.1 and sets thm.temp100 to the temperature multiplied by 100 and thm.tempReason to MEAS_DONE. If the value is out of scope, sets thm.temp100 to OUT_OF_RANGE and thm.tempReason to OUT_OF_LOWER_LIM or OUT_OF_UPPER_LIM. |
| getHum | Creates a humidity table at concerned temperature based on thm.temp100, calculates the humidity based on the sensor count measured using RFC Ch.0 and sets thm.hum10 to the humidity multiplied by 10 and thm.humReason to MEAS_DONE. If the value is out of scope, sets thm.hum10 to OUT_OF_RANGE. Also, if the temperature is out of range, sets thm.humReason to OUT_OF_TEMP_RANGE. If the humidity is out of range even if the temperature is within range, sets thm.humReason to OUT_OF_LOWER_LIM or OUT_OF_UPPER_LIM. |
| setHumMeasCount | Stores measured count indicated by thm.measCountIndex0 in thm.measCount0. |
| changeHumMeasCount | Adjusts the value of thm.measCountIndex0 so that the value of thm.sensorCount0 can be within range. |

### 4.2.3 Structure

This section describes structures defined in the sample programs.

Structure name: THERM_HYGRO_METER (file name: src/rfc.h)

| Variable Name | Type | Function |
|---|---|---|
| stat0 | unsigned short | Status of RFC Ch.0. |
| stat1 | unsigned short | Status of RFC Ch.1. |
| temp100 | signed short | Temperature which is calculated from the sensor count measured using RFC Ch.1 and multiplied by 100. |
| hum10 | unsigned short | A humidity table is created at concerned temperature based on temp100 of the same structure. The value is calculated from the sensor count measured using RFC Ch.0 and multiplied by 10. |
| tempReason | unsigned short | OUT_OF_LOWER_LIM (lower temperature than lower limit) or OUT_OF_UPPER_LIM (higher temperature than higher limit) if temp100 of the same structure is OUT_OF_RANGE. |
| humReason | unsigned short | OUT_OF_TEMP_RANGE (temperature out of range), OUT_OF_LOWER_LIM (lower humidity than lower limit), or OUT_OF_UPPER_LIM (higher humidity than higher limit) if hum10 of the same structure is OUT_OF_RANGE. |
| measCountIndex0 | unsigned short | Index of initial value (0x000000-n) for measurement counter for putting it in RFC Ch.0. |
| measCount0 | unsigned long | Initial value (0x000000-n) for measurement counter for putting it in RFC Ch.0. |
| sensorCount0 | unsigned long | Sensor count measured using RFC Ch.0. |
| sensorCount1 | unsigned long | Sensor count measured using RFC Ch.1. |

Structure name: THERM_HYGRO_TABLE (file name: src/rfc_tbl.h)

| Variable Name | Type | Function |
|---|---|---|
| dThermTemp100 | unsigned short | THERM_TEMP_STEP (temperature intervals of array thermSensCount) multiplied by 100. |
| dHygroTemp10 | unsigned short | HYGRO_TEMP_STEP (temperature intervals of array hygroCoef) multiplied by 10. |
| dHygroHum10 | unsigned short | HYGRO_HUM_STEP (humidity interval of array hygroCoef) multiplied by 10. |
| thermTemp100 | signed short | Each temperature of array thermSensCount multiplied by 100. |
| dThermSensCount | unsigned short | Difference (proportional to slope) between counter values of array thermSensCount. |
| hygroTemp10 | signed short | Each temperature of array hygroCoef multiplied by 10. |
| hygroHum10 | unsigned short | Each humidity of array hygroCoef multiplied by 10. |
| newHygroCoef | unsigned long | Humidity table which is corrected based on measured temperature thm.temp100 and created from array hygroCoef. |

### 4.2.4 Global Variable

This section describes global variables used in the sample programs.

File name: src/rfc.c

| Variable Name | Type | Function |
|---|---|---|
| thm | THERM_HYGRO_METER | Structure variable to store calculation results and measurement conditions for temperatures and humidities obtained from RFC Ch.0 and Ch.1. |

File name: src/rfc_tbl.c

| Variable Name | Type | Function |
|---|---|---|
| tht | THERM_HYGRO_TABLE | Structure variable to store the values which are used for accelerating the calculation when converting measured values obtained using RFC Ch.0 and Ch.1 into temperature and humidity. |

### 4.2.5 Operating Procedures

The sample software includes two projects for GNU17 Version 2 (hereinafter referred to as GNU17v2) and GNU17 Version 3 (hereinafter referred to as GNU17v3), and two header file sets for S1C17W22 and S1C17W23.

Before the sample software for GNU17v2 or GNU17v3 can be used, configure the target model by following the procedure shown below.

(1) Copy the header files for the target model to be used to the inc folder of the sample software.

Example: If the target model is S1C17W23, copy the c17w23_reg.h file and the reg folder to the inc subfolder in the s1c17w22_w23_rfc_therm_hygro_meter_gnu17vx folder of the sample software.

(2) Select [Import...] from the [File] menu to start the Import Wizard, then select

[General] > [Existing Project to Workspace] (GNU17v2) or

[General] > [Existing Projects into Workspace] (GNU17v3).

(3) Select the project folder that contains the sample program:

s1c17w22_w23_rfc_therm_hygro_meter_gnu17v2 folder (GNU17v2) or

s1c17w22_w23_rfc_therm_hygro_meter_gnu17v3 folder (GNU17v3).

(4) Select [Copy projects into workspace] and then click the [Finish] button to exit the Import Wizard.

(5) Change the target CPU.

(GNU17v2)

1. Select the imported project in the [C/C++ project] view and select [Properties] from the [Project] menu.
2. Select [GNU17 General] from the property list in the [Properties] dialog box that appears.
3. Select the target CPU from the [Target CPU Device] drop-down list.
4. Click the [Apply] button.

(GNU17v3)

1. Select the imported project in the [Project Explorer] view and select [Properties] from the [Project] menu.
2. Select [GNU17 Setting] from the property list in the [Properties] dialog box that appears.
3. Select the target CPU from the [Target CPU] drop-down list.
4. Click the [OK] button to close the dialog box. Then, go to Step (7).

(6) Set the debugger's startup options. (GNU17v2 only)

1. Select [GNU17 GDB Commands] from the property list.
2. Click the [Create commands from template] button to display the [Create a simple startup command] dialog box.
3. Select "ICD Mini" from the [Debugger:] drop-down list and select [Execute flash ROM writing].
4. Click the [Overwrite] button to close the dialog box. Close the [Properties] dialog box as well.

(7) Edit the init.h header file that exists in the src folder of the project to change the target model defined.

Example: When the target is S1C17W23

```
//#define MCUSEL_C17W22      /// S1C17W22
#define MCUSEL_C17W23        /// S1C17W23
```

(8) Build the project.

Use IDE to build the s1c17w22_w23_rfc_therm_hygro_meter_gnu17vx project.

(9) Connect the equipment and turn on the power.

1. Connect SVT17W23, ICDmini, USB cable, and PC with each other.
2. Reset SVT17W23 and ICDmini.

(10) Run the sample software.

Use IDE to run the s1c17w22_w23_rfc_therm_hygro_meter_gnu17vx project.

### 4.2.6 Sample Program Operation Overview

(1) Initialize structure variable tht and interrupt level.

(2) Start to oscillate OSC1 (32.768kHz).

(3) Initialize LCD and clear the display.

(4) Switch the system clock from IOSC to OSC1 and stop IOSC.

(5) Initialize T16 Ch.0, RFC Ch.0, and RFC Ch.1.

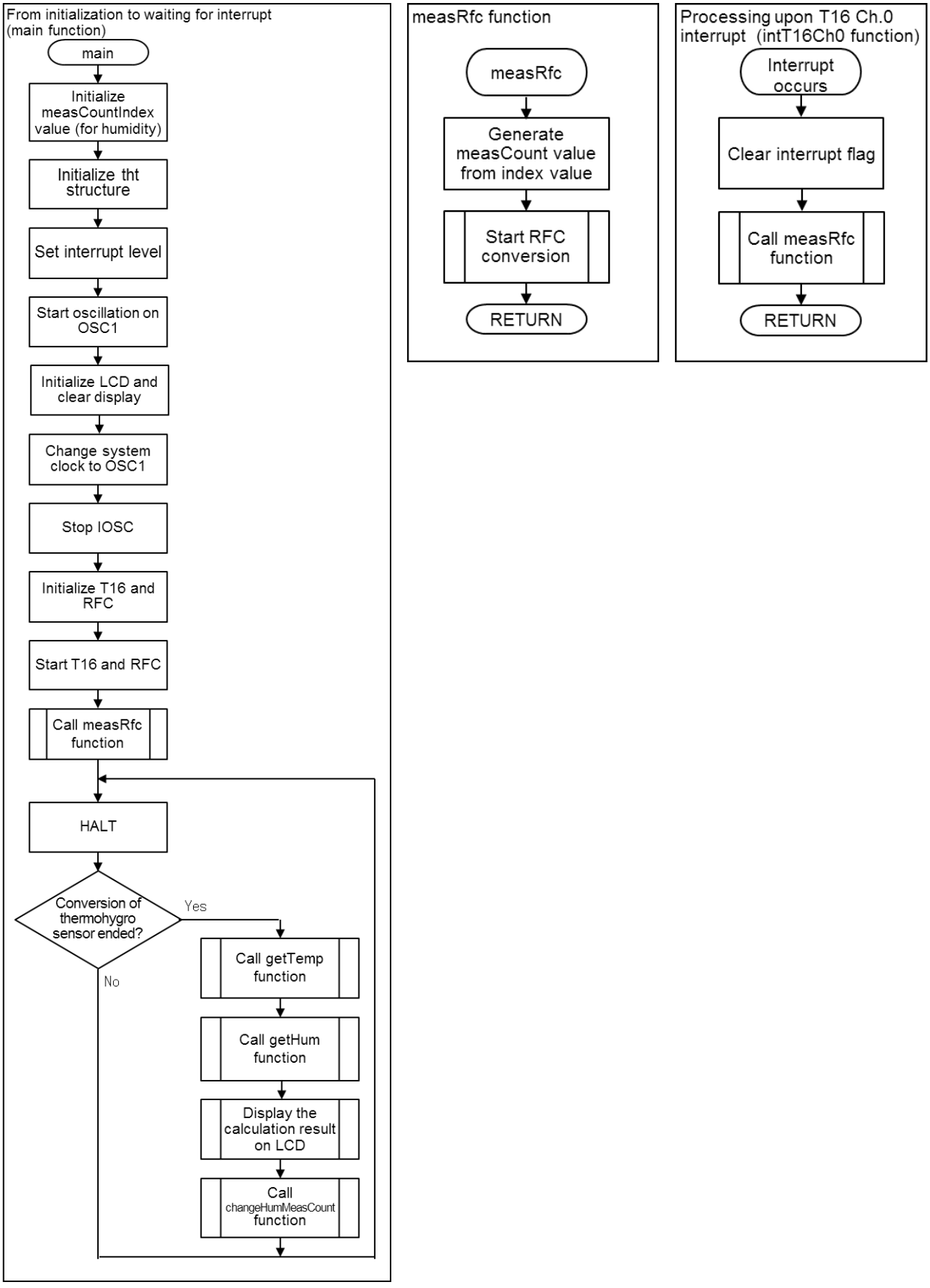
(6) Start to count T16 Ch.0, launch RFC Ch.0 and RFC Ch.1, and call the measurement startup function.

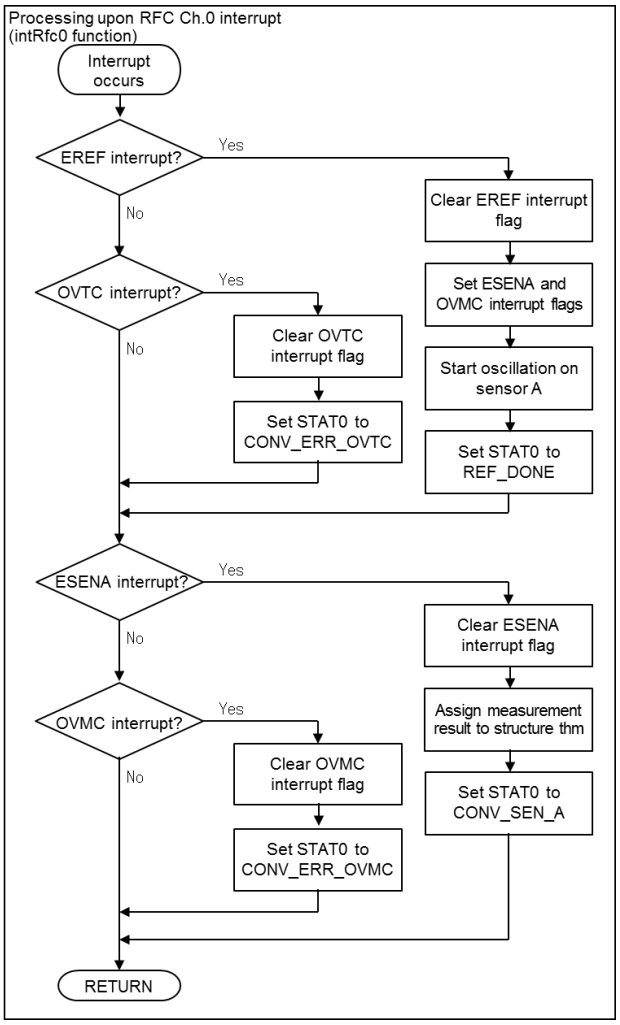
(7) The program halts to wait for an interrupt from T16 Ch0, RFC Ch.0, or RFC Ch.1.

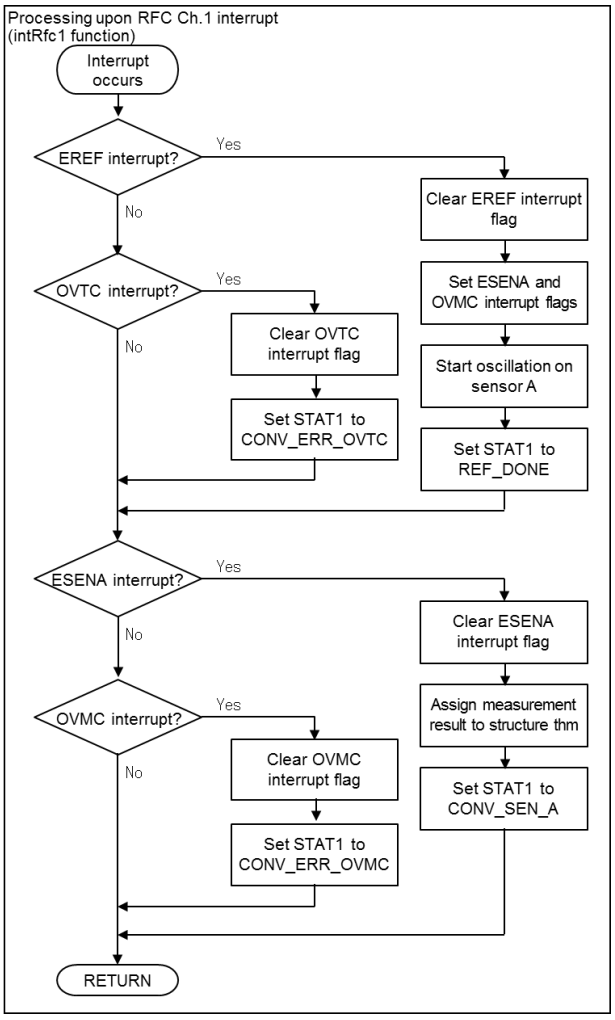
(8) When both RFC Ch.0 and RFC Ch.1 were measured, calculate the temperature and humidity, and display the results on LCD. Then, call the function changeHumMeasCount, adjust the thm.measCountIndex0 value so that the thm.sensorCount0 value can be within range, and perform step 7 and halt.

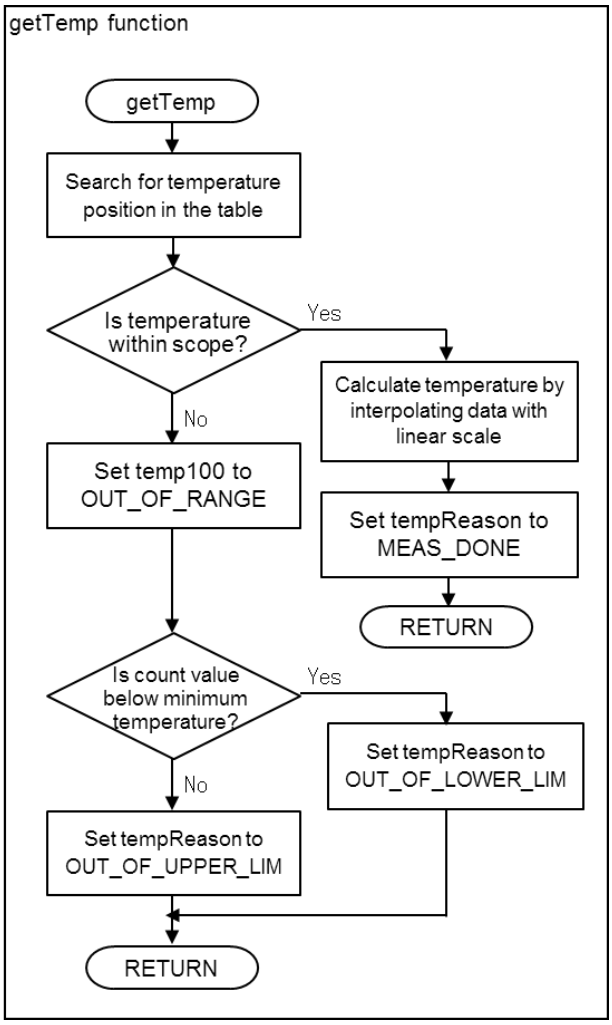
(9) Call the measurement startup function each time the program is interrupted by T16 Ch.0.

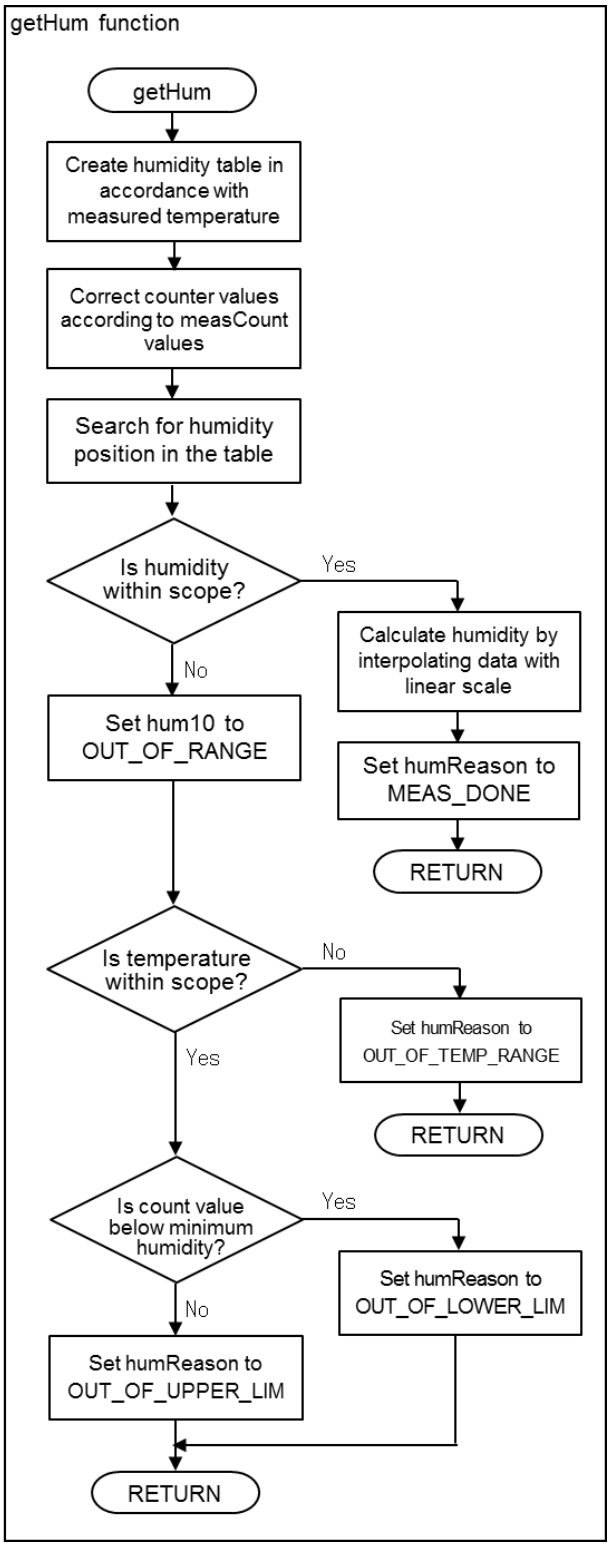
The figure below shows the flow chart:




Processing upon RFC Ch.0 interrupt
(intRfc0 function)
Interrupt occurs
EREF interrupt?
Yes
No
Clear EREF interrupt flag
Set ESENA and OVMC interrupt flags
Start oscillation on sensor A
Set STAT0 to REF_DONE
OVTC interrupt?
Yes
No
Clear OVTC interrupt flag
Set STAT0 to CONV_ERR_OVTC
ESENA interrupt?
Yes
No
Clear ESENA interrupt flag
Assign measurement result to structure thm
Set STAT0 to CONV_SEN_A
OVMC interrupt?
Yes
No
Clear OVMC interrupt flag
Set STAT0 to CONV_ERR_OVMC
RETURN


Processing upon RFC Ch.1 interrupt
(intRfc1 function)
Interrupt occurs
EREF interrupt?
Yes
No
Clear EREF interrupt flag
Set ESENA and OVMC interrupt flags
Start oscillation on sensor A
Set STAT1 to REF_DONE
OVTC interrupt?
Yes
No
Clear OVTC interrupt flag
Set STAT1 to CONV_ERR_OVTC
ESENA interrupt?
Yes
No
Clear ESENA interrupt flag
Assign measurement result to structure thm
Set STAT1 to CONV_SEN_A
OVMC interrupt?
Yes
No
Clear OVMC interrupt flag
Set STAT1 to CONV_ERR_OVMC
RETURN


getTemp function
getTemp
Search for temperature position in the table
Is temperature within scope?
Yes
No
Calculate temperature by interpolating data with linear scale
Set tempReason to MEAS_DONE
RETURN
Set temp100 to OUT_OF_RANGE
Is count value below minimum temperature?
Yes
No
Set tempReason to OUT_OF_LOWER_LIM
Set tempReason to OUT_OF_UPPER_LIM
RETURN


getHum function
getHum
Create humidity table in accordance with measured temperature
Correct counter values according to measCount values
Search for humidity position in the table
Is humidity within scope?
Yes
No
Calculate humidity by interpolating data with linear scale
Set humReason to MEAS_DONE
RETURN
Set hum10 to OUT_OF_RANGE
Is temperature within scope?
No
Yes
Set humReason to OUT_OF_TEMP_RANGE
RETURN
Is count value below minimum humidity?
Yes
No
Set humReason to OUT_OF_LOWER_LIM
Set humReason to OUT_OF_UPPER_LIM
RETURN

## 4.3 About s1c17w22_w23_rfc_saltConc_meter_gnu17vx

This section describes the software, s1c17w22_w23_rfc_saltConc_meter_gnu17vx, which converts RFC measured counts into salinity concentrations and temperatures and displays them on LCD. The RFC measured counts change according to the change of resistances between electrodes and resistances of temperature sensor.

### 4.3.1 File Configuration

| Filename | Function |
|---|---|
| src/boot.c | Startup module file |
| src/clg.c | CLG driver file |
| src/display.c | LCD display function file |
| src/init.c | Initialization function file |
| src/main.c | Main function file |
| src/rfc_tbl.c | Conversion table of RFC salinity concentration and temperature, function file |
| src/rfc.c | RFC driver file |
| src/svd.c | SVD driver file |
| src/t16_ch0.c | T16 Ch.0 driver file |
| src/crt0.h | Boot processing definition file (for GNU17 Version 3) |
| src/clg.h | CLG driver header definition file |
| src/display.h | LCD display function header definition file |
| src/init.h | Initialization function header definition file |
| src/lcd_font.h | LCD display font definition file |
| src/rfc_tbl.h | Conversion header definition file of RFC salinity concentration and temperature |
| src/rfc.h | RFC driver header definition file |
| src/svd.h | SVD driver header definition file |
| src/t16_ch0.h | T16 Ch.0 driver header definition file |
| inc/reg | Folder for storing peripheral device register definition files |
| inc/c17w22_reg.h<br>inc/c17w23_reg.h | Peripheral device header definition files |

### 4.3.2 Module Description

This section mainly describes non-generic function names, their functions, array variable names, and their details. (However, the description of the same contents as the programming package s1c17w22_w23_rfc_therm_hygro_meter_gnu17vx is omitted.)

File name: main.c

| Function Name | Function |
|---|---|
| initLcd | Initializes LCD mounted on SVT17W23. |
| measRfc | Starts the conversion using RFC Ch.0 and RFC Ch.1. |
| intT16Ch0 | T16 Ch.0 interrupt function. Calls the function measRfc above. |
| val2Str3 | Converts the value of unsigned short type into up to three digits of numeric value and substitutes the value into specified position of string. |
| dispVal | Displays the salinity concentration, temperature, and power supply voltage on LCD. When the value is out of scope, displays accordingly. |

File name: rfc_tbl.c

| Array Variable Name | Description |
|---|---|
| thermSensCount | Count value of the temperature sensor in the range from 0°C to 100°C at 2°C intervals. Measured value of RF count when n is 5000 (=THERM_MEAS_COUNT) which is used in initial value (0x000000-n) for measurement counter for RFC Ch.1. |
| salconCoef | Count value of measurement electrodes (salinity concentration sensor) in the range from 5°C to 85°C at 20°C intervals and in the range from 0.40% to 1.40% at 0.25% intervals when Vdd=3.3V. Measured value of RF count when n is 1000 (=SALCON_MEAS_COUNT) which is used in initial values (0x000000-n) for measurement counter for RFC Ch.0. |

File name: rfc_tbl.c

| Function Name | Function |
|---|---|
| initRfcTbl | Initializes the array elements of global structure variable tst. |
| getTemp | Calculates the temperature based on the sensor count measured using RFC Ch.1 and sets tsm.temp100 to the temperature multiplied by 100 and tsm.tempReason to MEAS_DONE. If the value is out of scope, sets tsm.temp100 to OUT_OF_RANGE and tsm.tempReason to OUT_OF_LOWER_LIM or OUT_OF_UPPER_LIM. |
| getConc | Sets tsm.svdVolt to the value according to the power supply voltage measured using SVD, creates the salinity concentration table for concerned temperature based on tsm.temp100, calculates the salinity concentrations based on the sensor counts measured using RFC Ch.0, and sets tsm.conc1000 to the salinity concentration in percentage multiplied by 1000 and tsm.concReason to MEAS_DONE. If the value is out of scope, sets tsm.hum1000 to OUT_OF_RANGE. Also, if the temperature is out of range, sets tsm.concReason to OUT_OF_TEMP_RANGE. If the salinity concentration is out of range even if the temperature is within range, sets tsm.concReason to OUT_OF_LOWER_LIM or OUT_OF_UPPER_LIM. |

### 4.3.3 Structure

This section describes structures defined in the sample programs.

Structure name: THERM_SALCON_METER (file name: src/rfc.h)

| Variable Name | Type | Function |
|---|---|---|
| stat0 | unsigned short | Status of RFC Ch.0. |
| stat1 | unsigned short | Status of RFC Ch.1. |
| temp100 | signed short | Temperature which is calculated from the sensor count measured using RFC Ch.1 and multiplied by 100. |
| conc1000 | unsigned short | A salinity concentration table is created at concerned temperature based on temp100 of the same structure. The salinity concentration in percentage is calculated from the sensor count measured using RFC Ch.0 and multiplied by 1000. |
| svdVolt | unsigned short | Power supply voltage measured using SVD displayed in the format of SVDCTL_SVDC_XXXXMV ("XXXX" represents model symbols) defined in inc/svd.h. |
| tempReason | unsigned short | OUT_OF_LOWER_LIM (lower temperature than lower limit) or OUT_OF_UPPER_LIM (higher temperature than higher limit) if temp100 of the same structure is OUT_OF_RANGE. |
| concReason | unsigned short | OUT_OF_TEMP_RANGE (temperature out of range), OUT_OF_LOWER_LIM (lower temperature than lower limit), or OUT_OF_UPPER_LIM (higher temperature than higher limit) if conc1000 of the same structure is OUT_OF_RANGE. |
| sensorCount0 | unsigned long | Sensor count measured using RFC Ch.0. |
| sensorCount1 | unsigned long | Sensor count measured using RFC Ch.1. |

Structure name: THERM_SALCON_TABLE (file name: src/rfc_tbl.h)

| Variable Name | Type | Function |
|---|---|---|
| dThermTemp100 | unsigned short | THERM_TEMP_STEP (temperature intervals of array thermSensCount) multiplied by 100. |
| dSalconTemp10 | unsigned short | SALCON_TEMP_STEP (temperature intervals of array salconCoef) multiplied by 10. |
| dSalconConc1000 | unsigned short | SALCON_CONC_STEP100 (salinity concentration intervals of array salconCoef in percentage multiplied by 100) multiplied by 10. |
| thermTemp100 | signed short | Each temperature of array thermSensCount multiplied by 100. |
| dThermSensCount | unsigned short | Difference (proportional to slope) between counter values of array thermSensCount. |
| salconTemp10 | signed short | Each temperature of array salconCoef multiplied by 10. |
| salconConc1000 | unsigned short | Each salinity concentration in percentage of array salconCoef multiplied by 1000. |
| newSalconCoef | unsigned long | Salinity concentration table which is corrected based on measured temperature tsm.temp100 and created from array salconCoef. |

### 4.3.4 Global Variable

This section describes global variables used in the sample programs.

File name: src/rfc.c

| Variable Name | Type | Function |
|---|---|---|
| tsm | THERM_SALCON_METER | Structure variable to store calculation results and measurement conditions for salinity concentration and temperature obtained from RFC Ch.0 and Ch.1. |

File name: src/rfc_tbl.c

| Variable Name | Type | Function |
|---|---|---|
| tst | THERM_SALON_TABLE | Structure variable to store the values which are used for accelerating the calculation when converting measured values obtained from RFC Ch.0 and Ch.1 into salinity concentrations and temperatures. |

### 4.3.5 Operating Procedures

The sample software includes two projects for GNU17 Version 2 (hereinafter referred to as GNU17v2) and GNU17 Version 3 (hereinafter referred to as GNU17v3), and two header file sets for S1C17W22 and S1C17W23.

Before the sample software for GNU17v2 or GNU17v3 can be used, configure the target model by following the procedure shown below.

(1) Copy the header files for the target model to be used to the inc folder of the sample software.

Example: If the target model is S1C17W23, copy the c17w23_reg.h file and the reg folder to the inc subfolder in the s1c17w22_w23_rfc_saltConc_meter_gnu17vx folder of the sample software.

(2) Select [Import...] from the [File] menu to start the Import Wizard, then select

[General] > [Existing Project to Workspace] (GNU17v2) or

[General] > [Existing Projects into Workspace] (GNU17v3).

(3) Select the project folder that contains the sample program:

s1c17w22_w23_rfc_saltConc_meter_gnu17v2 folder (GNU17v2) or

s1c17w22_w23_rfc_saltConc_meter_gnu17v3 folder (GNU17v3).

(4) Select [Copy projects into workspace] and then click the [Finish] button to exit the Import Wizard.

(5) Change the target CPU.

(GNU17v2)

1. Select the imported project in the [C/C++ project] view and select [Properties] from the [Project] menu.
2. Select [GNU17 General] from the property list in the [Properties] dialog box that appears.
3. Select the target CPU from the [Target CPU Device] drop-down list.
4. Click the [Apply] button.

(GNU17v3)

1. Select the imported project in the [Project Explorer] view and select [Properties] from the [Project] menu.
2. Select [GNU17 Setting] from the property list in the [Properties] dialog box that appears.
3. Select the target CPU from the [Target CPU] drop-down list.
4. Click the [OK] button to close the dialog box. Then, go to Step (7).

(6) Set the debugger's startup options. (GNU17v2 only)

1. Select [GNU17 GDB Commands] from the property list.
2. Click the [Create commands from template] button to display the [Create a simple startup command] dialog box.
3. Select "ICD Mini" from the [Debugger:] drop-down list and select [Execute flash ROM writing].
4. Click the [Overwrite] button to close the dialog box. Close the [Properties] dialog box as well.

(7) Edit the init.h header file that exists in the src folder of the project to change the target model defined.

Example: When the target is S1C17W23

```
//#define MCUSEL_C17W22      /// S1C17W22
#define MCUSEL_C17W23        /// S1C17W23
```

(8) Build the project.

Use IDE to build the s1c17w22_w23_rfc_saltConc_meter_gnu17vx project.

(9) Connect the equipment and turn on the power.

1. Connect SVT17W23, ICDmini, USB cable, and PC with each other.
2. Reset SVT17W23 and ICDmini.

(10) Run the sample software.

Use IDE to run the s1c17w22_w23_rfc_saltConc_meter_gnu17vx project.

### 4.3.6 Sample Program Operation Overview

(1) Initialize structure variable tst and interrupt level.

(2) Start to oscillate OSC1 (32.768kHz).

(3) Initialize LCD and clear the display.

(4) Switch the system clock from IOSC to OSC1 and stop IOSC.

(5) Initialize T16 Ch.0, SVD, RFC Ch.0, and RFC Ch.1.

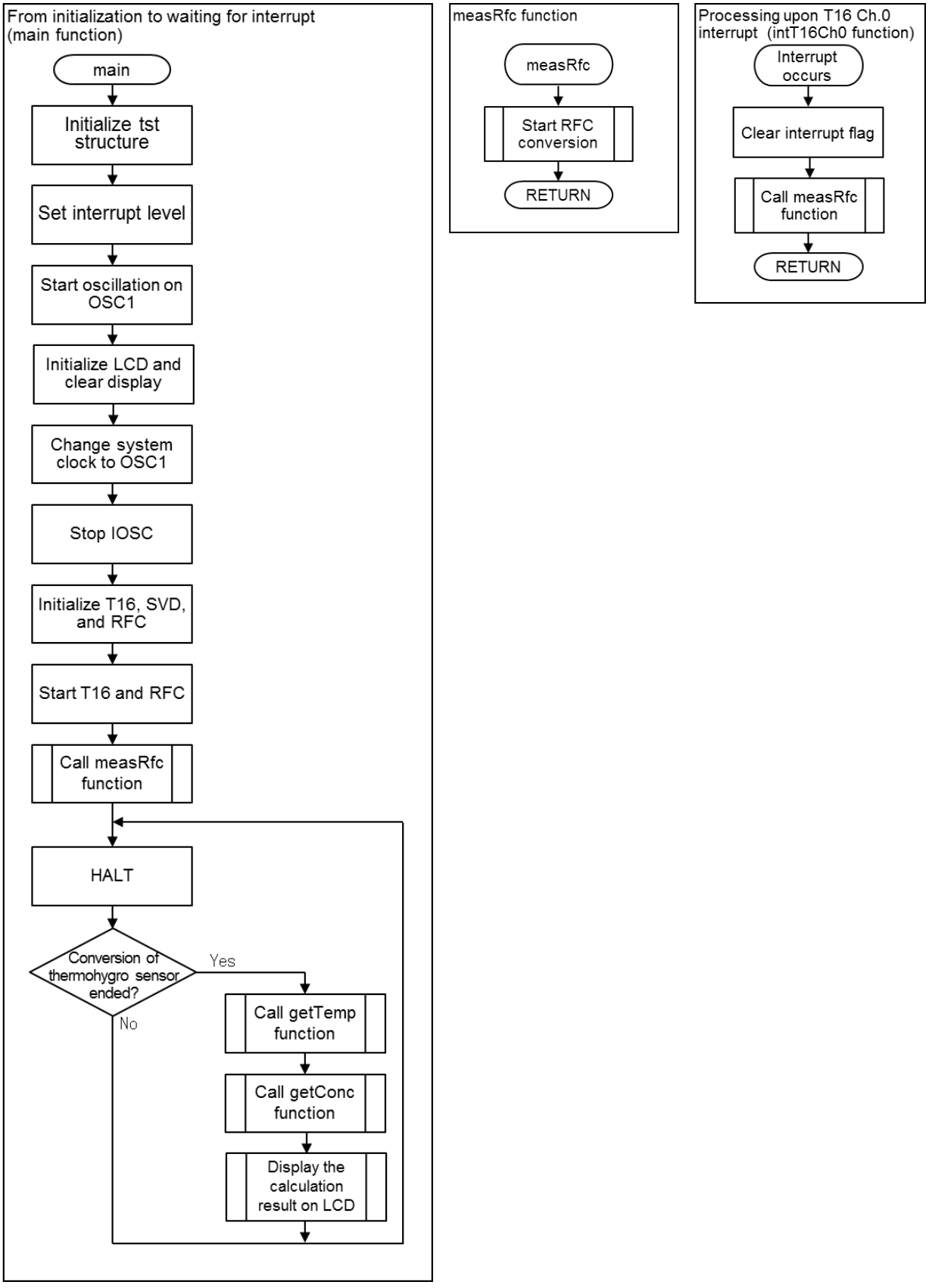
(6) Start to count T16 Ch.0, launch RFC Ch.0 and RFC Ch.1, and call the measurement startup function.

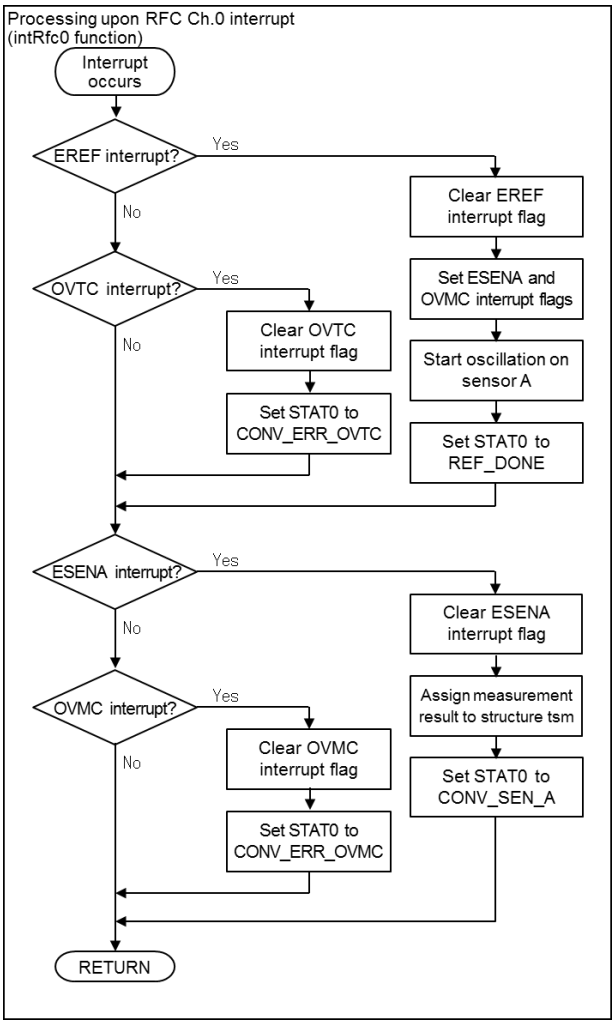
(7) The program halts to wait for an interrupt from T16 Ch0, RFC Ch.0, or RFC Ch.1.

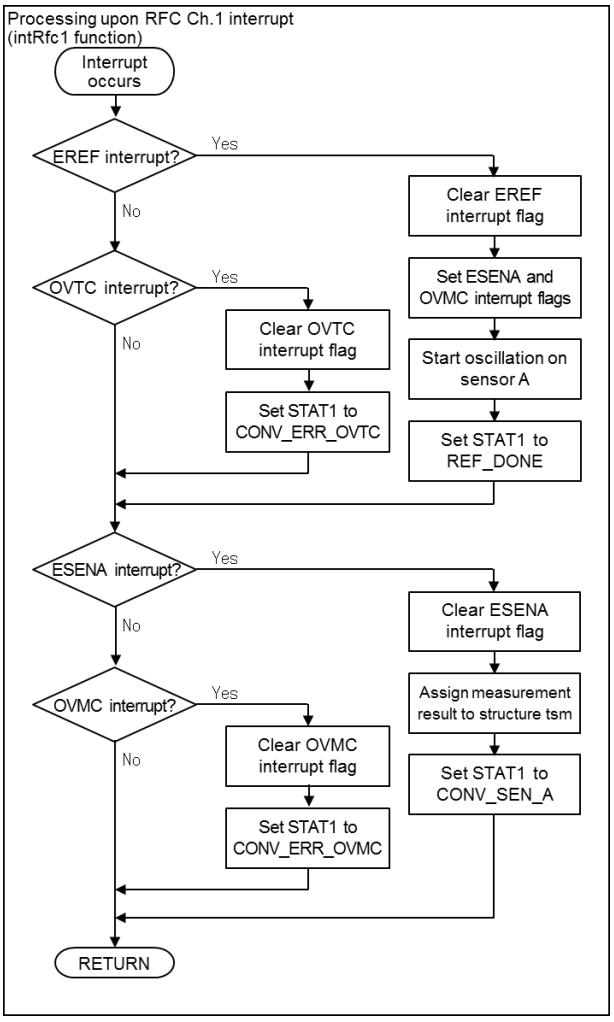
(8) When both RFC Ch.0 and RFC Ch.1 were measured, calculate the salinity concentrations and temperatures, and display the results as well as the power supply voltages detected using SVD on LCD. Then, perform step 7 and halt.

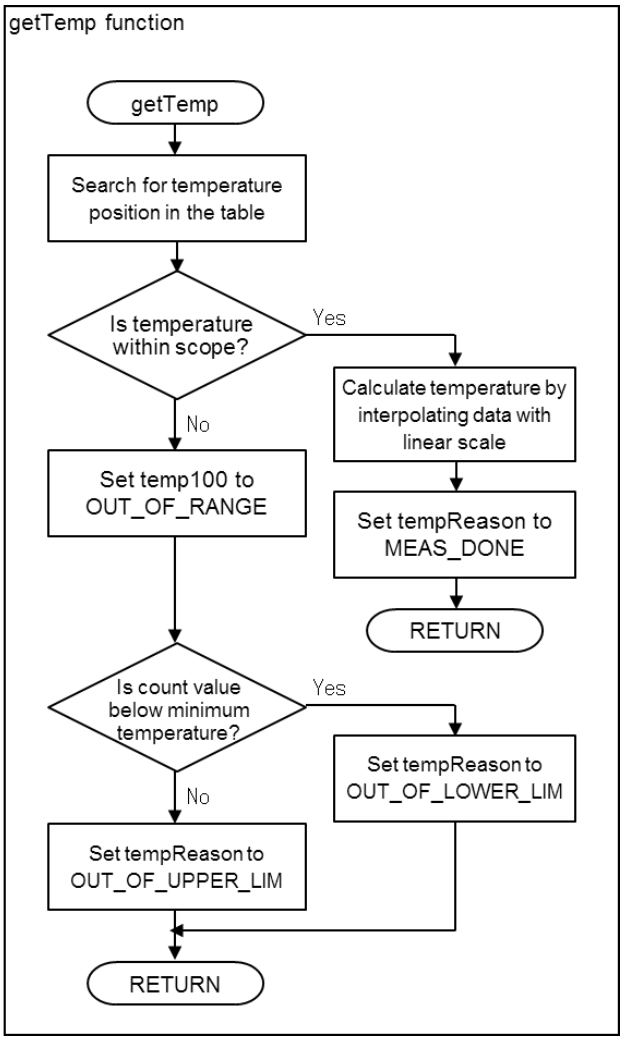
(9) Call the measurement startup function each time the program is interrupted by T16 Ch.0.

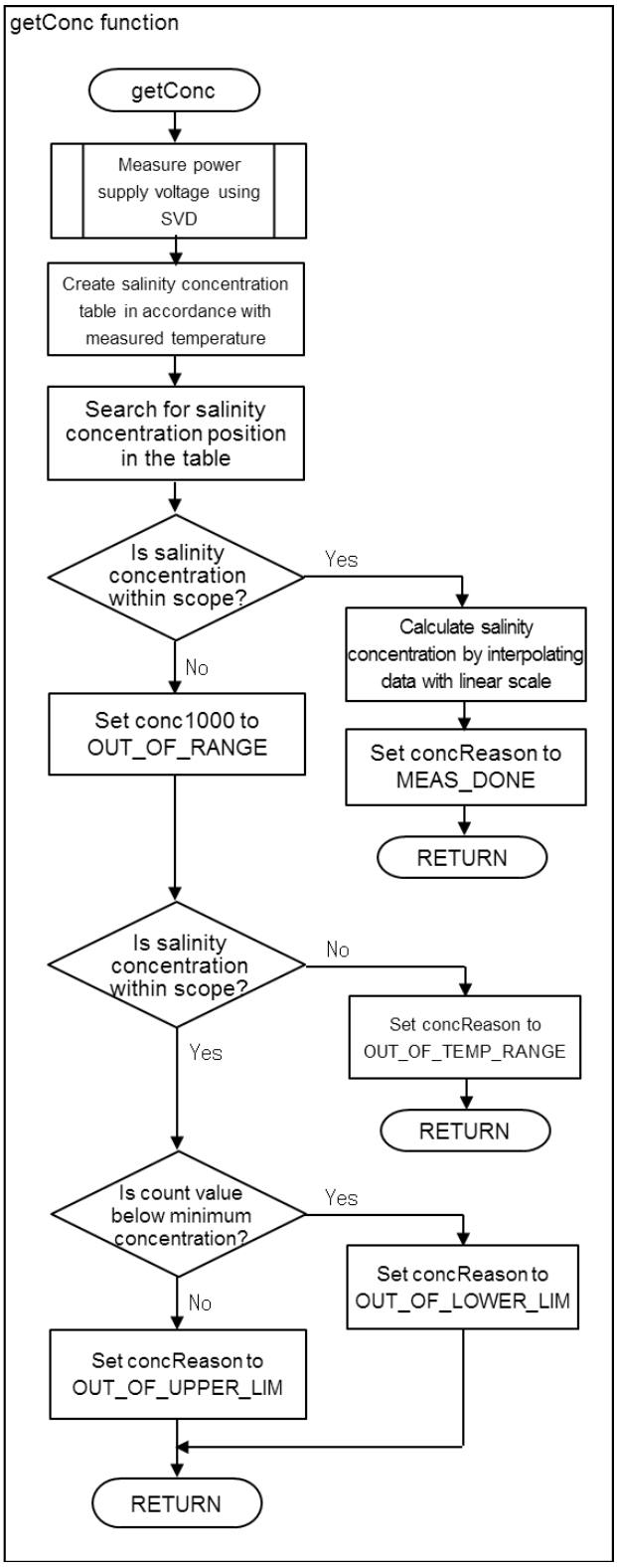
The figure below shows the flow chart:




Processing upon RFC Ch.0 interrupt
(intRfc0 function)
Interrupt occurs
EREF interrupt?
Yes
No
Clear EREF interrupt flag
Set ESENA and OVMC interrupt flags
Start oscillation on sensor A
Set STAT0 to REF_DONE
OVTC interrupt?
Yes
No
Clear OVTC interrupt flag
Set STAT0 to CONV_ERR_OVTC
ESENA interrupt?
Yes
No
Clear ESENA interrupt flag
Assign measurement result to structure tsm
Set STAT0 to CONV_SEN_A
OVMC interrupt?
Yes
No
Clear OVMC interrupt flag
Set STAT0 to CONV_ERR_OVMC
RETURN


Processing upon RFC Ch.1 interrupt
(intRfc1 function)
Interrupt occurs
EREF interrupt?
Yes
No
Clear EREF interrupt flag
Set ESENA and OVMC interrupt flags
Start oscillation on sensor A
Set STAT1 to REF_DONE
OVTC interrupt?
Yes
No
Clear OVTC interrupt flag
Set STAT1 to CONV_ERR_OVTC
ESENA interrupt?
Yes
No
Clear ESENA interrupt flag
Assign measurement result to structure tsm
Set STAT1 to CONV_SEN_A
OVMC interrupt?
Yes
No
Clear OVMC interrupt flag
Set STAT1 to CONV_ERR_OVMC
RETURN


getTemp function
getTemp
Search for temperature position in the table
Is temperature within scope?
Yes
No
Calculate temperature by interpolating data with linear scale
Set tempReason to MEAS_DONE
RETURN
Set temp100 to OUT_OF_RANGE
Is count value below minimum temperature?
Yes
No
Set tempReason to OUT_OF_LOWER_LIM
Set tempReason to OUT_OF_UPPER_LIM
RETURN


getConc function
getConc
Measure power supply voltage using SVD
Create salinity concentration table in accordance with measured temperature
Search for salinity concentration position in the table
Is salinity concentration within scope?
Yes
No
Calculate salinity concentration by interpolating data with linear scale
Set concReason to MEAS_DONE
RETURN
Set conc1000 to OUT_OF_RANGE
Is salinity concentration within scope?
No
Yes
Set concReason to OUT_OF_TEMP_RANGE
RETURN
Is count value below minimum concentration?
Yes
No
Set concReason to OUT_OF_LOWER_LIM
Set concReason to OUT_OF_UPPER_LIM
RETURN

# APPENDIX A. Creating Example of Thermohygro Table

Here, describes an example how to create the "Conversion table between R/F converter count and temperature" and "Conversion table between R/F converter count and humidity." There are several ways to create the conversion table. One is to measure resistances of sensor under some conditions, create a resistance table by converting the resistances based on the sensor's data sheet, and measure count values by connecting each resistor with specified value to the sensor section of RFC. Another is to measure count values under several conditions actually by using sensor and thermostatic bath. This sample program used the latter.

The actual measurement used the circuits shown in Figures 3-2, 3-3 and the parts listed in Table 3-1 (BOM). Also, the experiment used the thermostatic bath below, but the results of this material should be seen only as reference.

- Thermostatic bath: ESPEC Temp. & Humid. Chamber SH-240

The sample program created this time has the following range for measuring temperatures and humidities as a goal.

Temperature: -20.0°C to 50.0°C, resolution: 0.1°C
Humidity: 10%rh to 90%rh, resolution: 1%rh

### (1) Confirming sensor's range of resistances

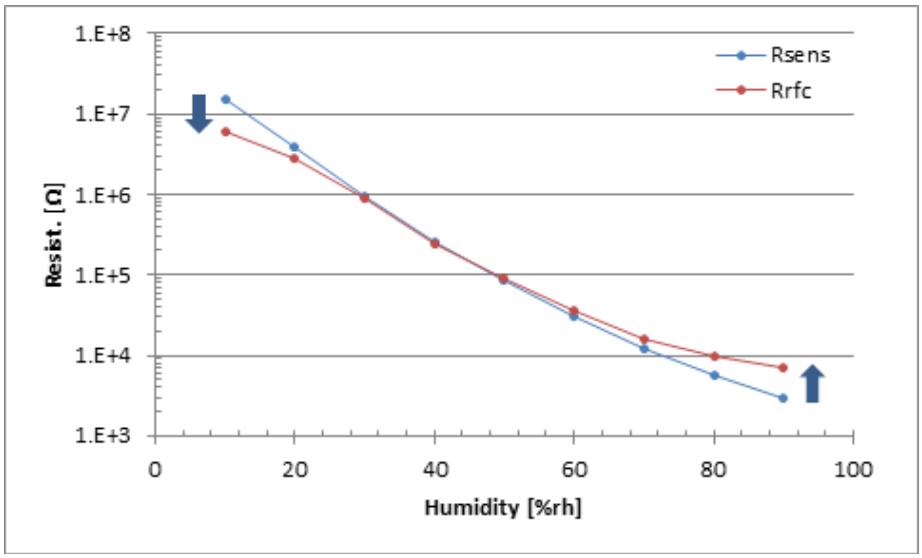
Confirm the minimum and maximum resistances (Rmin and Rmax) in desired range based on the sensor characteristics.

Thermistor: 103AT-2B manufactured by SEMITEC Corporation
Rmin(typ.)=10.18kΩ (at 50°C) Rmax(typ.)=69.01kΩ (at -20°C)

Humidity sensor: CL-M52R manufactured by Shinyei Technology
Rmin: around 3kΩ (at 90%rh, 25°C) Rmax: around 15MΩ (at 10%rh, 25°C)

### (2) Adjusting dynamic range using external resistor

Since the dynamic range of the change of resistances is wide for humidity sensor, add a parallel resistance R101 and a series resistance R102. When R101=10MΩ and R102=4.3kΩ, listed in Table 3-1 (BOM) are adopted in the circuit shown in Figure 3-2, the response characteristics change to narrow dynamic range, as shown in the range between Rsens and Rrfc below.



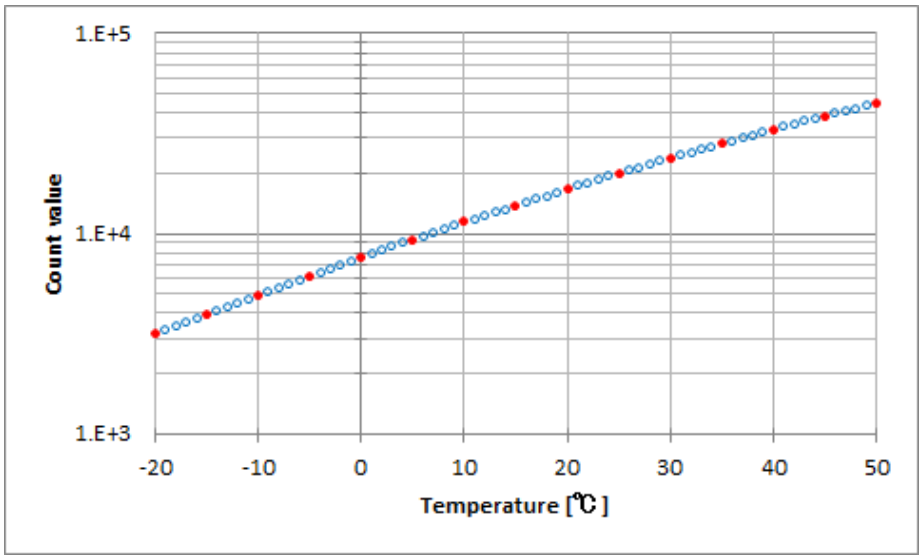
As described in electrical characteristics of R/F converter (RFC) in the technical manual, the reference resistance and resistance of resistive sensor should be 1kΩ or more. This specification also means that an error becomes bigger due to causes of other than sensor, such as the IC's production variation, when the resistance decreases to around 1kΩ. The slope of curve is made lower in high humidity area of the response characteristics above lest the resistance comes near 1kΩ.

Similarly, the resistance is made small at low humidity lest oscillating frequency becomes too low so that the measured counts decrease in prescribed time and the error of measured values increases.

### (3) Measuring dependency of count values on temperature for temperature sensor

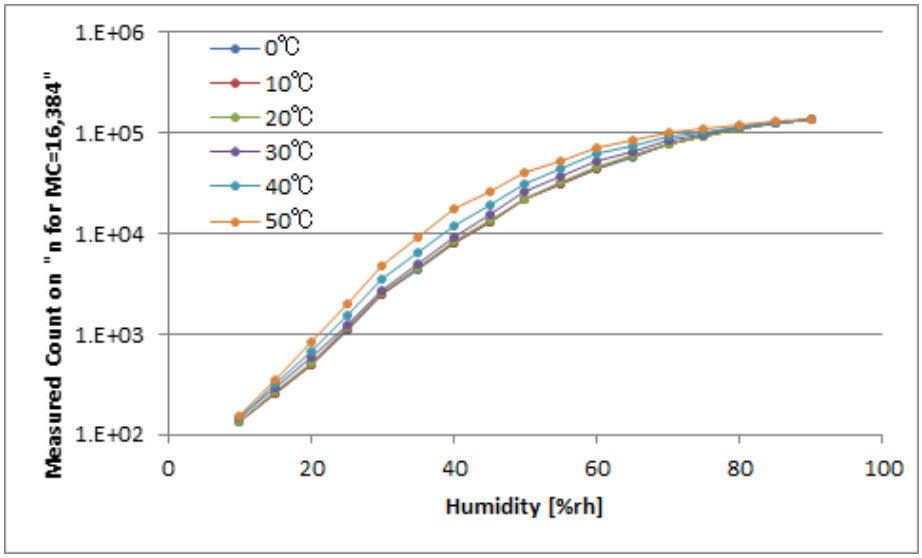
Use the thermostatic bath and then confirm the dependency of count values on temperature for temperature sensor. This time, the temperatures were measured in the range between -20°C and 50°C at 5°C intervals. The results were interpolated with logarithmic scale so that the intervals will be 1°C and then the count values were calculated. The red points in the graph below represent measured values and blue points represents interpolated values.

The reason why the intervals are decreased by adding table data in the interpolation is to diminish the error in the calculation using linear scale, as shown in graph (b) of Figure 3-4, when calculating temperatures in MCU.



### (4) Measuring dependency of count values on temperature and humidity for humidity sensor

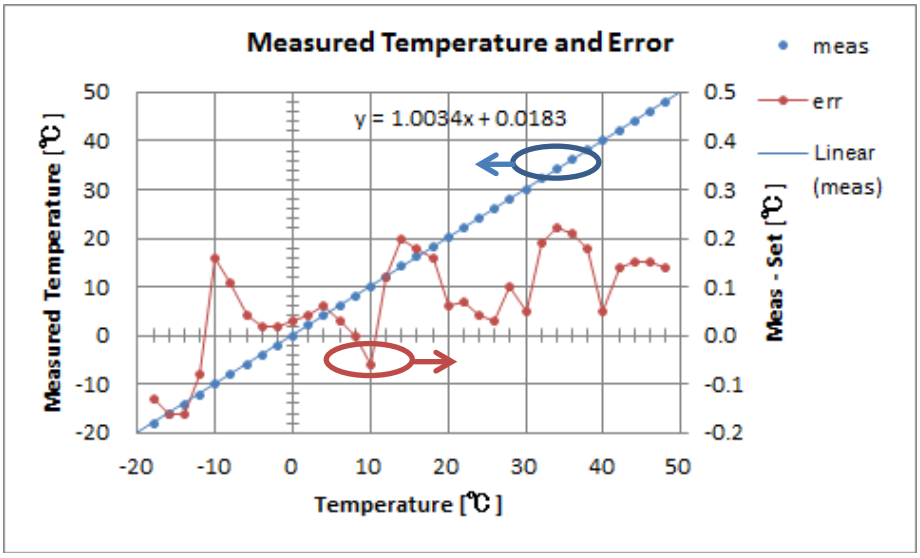
Use the thermostatic bath and then confirm the dependency of count values on temperature and humidity for humidity sensor. This time, the temperatures were measured in the range between 10°C and 50°C at 10°C intervals and the humidities were measured in the range between 30%rh and 90%rh at 10%rh intervals. The results were interpolated with logarithmic scale so that the intervals will be 5%rh and then the count values were calculated. The reason of the processing above is the same as temperature sensor. In low temperature and low humidity area which cannot be interpolated, the values are selected so that the slope of plots will be smooth. As the result above, in the combination of the range between 0°C and 50°C at 10°C intervals and the range between 10%rh and 90%rh at 5%rh intervals, the dependency of count values on temperature can be gotten for the humidity sensor.




### (5) Confirming error of temperature measurement result

The temperature is changed in the range between -20°C and 50°C at 2°C intervals by using the parameters obtained above and the measured temperatures are compared with setup temperature of thermostatic bath. Note that, as for measured temperature, the element thm.temp100 of structure thm divided by 100 was used.

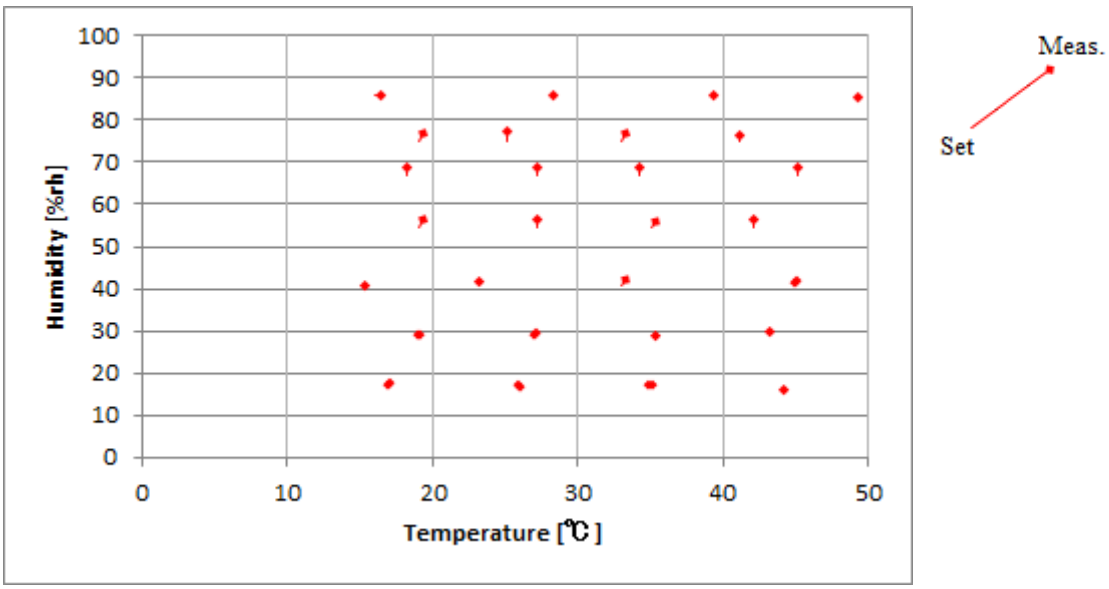
The graph below shows the results plotted. The horizontal axis is the setup temperature of thermostatic bath, the blue points represents measured temperatures by changing the temperature, the blue line represents linear approximation of temperatures, and the vertical axis of them is at the left. The red plots represent errors, measured temperatures subtracted by setup temperature, and the vertical axis of them is at the right.



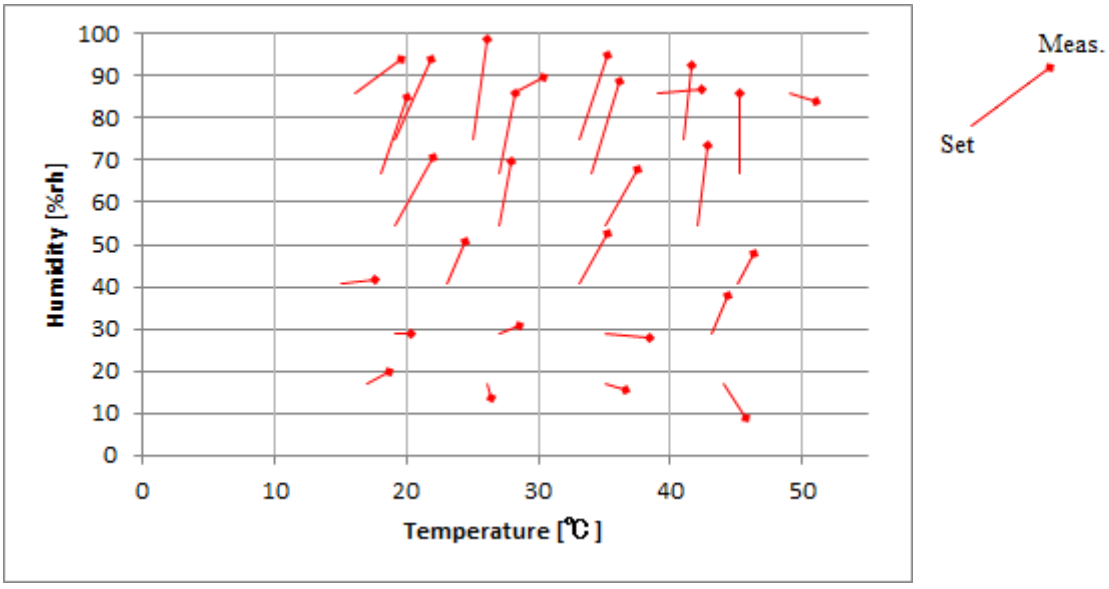
### (6) Confirming error of humidity measurement result

The temperature and humidity are measured at some points in the range between 15°C and 49°C and in the range between 17%rh and 86%rh by using the parameters obtained above and the measured temperatures and humidities are compared with setting values of thermostatic bath. Similarly, as for measured temperature, the element thm.temp100 of structure thm divided by 100 was used, and as for measured humidity, the element thm.hum10 divided by 10 was used.

The graph below shows the results plotted. The horizontal axis is temperature and the vertical axis is relative humidity. Making the analogy of a matchstick, the stick end represents the setting value and the head of the matchstick represents measured value.




In the graph above, since measured values are adjacent to setting values, the graph below is replotted by multiplying the error by 10 (multiply the vector length to the head by 10 while the stick end is at the same point as before).



### (7) Processing time

It is a processing time measured using the [c17 clock] command. It is only for your reference and it does not guarantee the processing time.

Measurement result: temperature: 24.4°C, humidity: 15%rh

| | |
|---|---|
| Measure temperature (runRfcConverintgOperation to intRfc1) | 134ms |
| Measure humidity (runRfcConveringOperation to intRfc0) | 510ms |
| Calculate temperature (running time of getTemp function) | 135ms |
| Calculate humidity (running time of getHum function) | 1070ms |
| Display result (running time of dispVal)) | 147ms |

Since the measurements are processed in parallel, the measurement ends when the process which takes longer time ends.

### (8) Range of measured count

In "3.3. How to Measure Resistance of Humidity Sensor", it is described as "The resistances of humidity sensor should be measured using RFC so that measured counts are within certain range." In Appendix A, the range is determined as follows: (The values are defined in src/rfc_tbl.c.)

| | |
|---|---|
| HUM_COUNTER_LOWER_LIM | 500 |
| HUM_COUNTER_UPPER_LIM | 5,000 |

HUM_COUNTER_UPPER_LIM=5,000 is because of the constraint condition: Measured count should not exceeds 10000. HUM_COUNTER_LOWER_LIM=500 is only because the value will not give any impact on the humidity resolution: 1%rh.

### (9) Impact of humidity on OSC1 oscillation

The oscillating circuit of S1C17 series OSC1 is designed to make oscillation only with really few electric power. Therefore, when the humidity rises, the oscillation may stop due to the impact from leak current.

When using OSC1 under usual living condition, it's possible to reduce the impact from leak current by changing the value of CLGOSC1.INVIN register and increasing the gain of oscillation inverter.

However, under significantly high humidity environment, it cannot avoid the impact perfectly. If this is the case, it's necessary to apply moisture-proof coating to the OSC1 system at least or whole system preferably.

# APPENDIX B. Creating Example of Salinity Concentration Conversion Table

Here, describes an example how to create the "Conversion table between R/F converter count and salinity concentration." Similar to humidity sensor, the conversion table was created by performing the way to measure count values under some conditions actually using electrodes for salinity concentration sensor and thermostatic bath.

The actual measurement used the circuits shown in Figures 3-2, 3-3 and the parts listed in Table 3-2 (BOM). Also, the experiment used the thermostatic bath below, but the results of this material should be seen only as reference. In general, it takes less time to reach a constant temperature by using water bath in place of thermostatic bath.

- Thermostatic bath: ESPEC Temp. & Humid. Chamber SH-240

The sample program created this time has the following range for measuring salinity concentrations as a goal.

Salinity concentration: 0.40% to 1.40%, resolution: 0.01%

Then, temperature range and power supply voltage are assumed to be in the following range and the value.

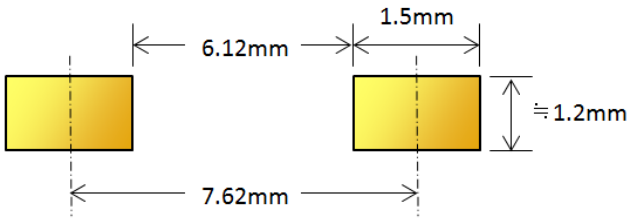
Temperature range: 5°C to 85°C (the table is in the range between 0°C and 100°C)
Power supply voltage: 3.3V

In the assumption above, the power supply voltage is set to constant 3.3V because the dependency of resistance between electrodes on temperature is relatively great according to preliminary experiments. It is also because to avoid the increase of parameters. The actual salinometer is assumed that the power is supplied to MCU by stabilizing about 4.5V of battery voltage to 3.3V with power supply regulator.

The way to measure the temperatures of solution is totally the same as temperature and humidity measurements except that the temperature range and intervals of the table used are different. For information on the procedures, refer to "APPENDIX A. Creating Example of Thermohygro Table."

### (1) Structure of electrodes for salinity concentration sensor

The following electrodes are used as a salinity concentration sensor.




Though round or spherical electrodes are preferable by taking into account concentrated electric field, this application notes used gold-plated rectangle copper foil. As for the product which actually measures the salinity concentrations of foods, in order to conform the Food Sanitation Law, the electrode material should not contain harmful metals to human body, such as lead, and the thickness of gold plate should be checked deliberately.

### (2) Estimated range of resistances of salinity concentration sensor

The range of resistances of the salinometer is 0.40% to 1.40%. The conductance σ of it may change in the range between 7.4mS/cm and 23mS/cm at 25°C. The rectangle electrodes above have the area A (=1.5mm x 1.2mm) and face each other a distance d (=7.62mm) apart. The range of resistances under these conditions is 5.7kΩ to 1.8kΩ from the expression below:

$$R = \frac{1}{\sigma} \cdot \frac{d}{A}$$

### (3) Adjusting resistance due to external resistance

As described in "3.4. Measuring Salinity Concentration using R/F Converter," lest the resistance of electrodes as a salinity concentration sensor is too low to satisfy the specification for electrical characteristics of the R/F converter ($R_{REF}$(min)=1kΩ) or in order to detect infinite resistance when the electrodes are not soaked in the solution and stay in the air, a parallel resistance R101 and a series resistance R102 are added. Their resistances are R101=1MΩ and R102=1kΩ according to Table 3-2 (BOM).

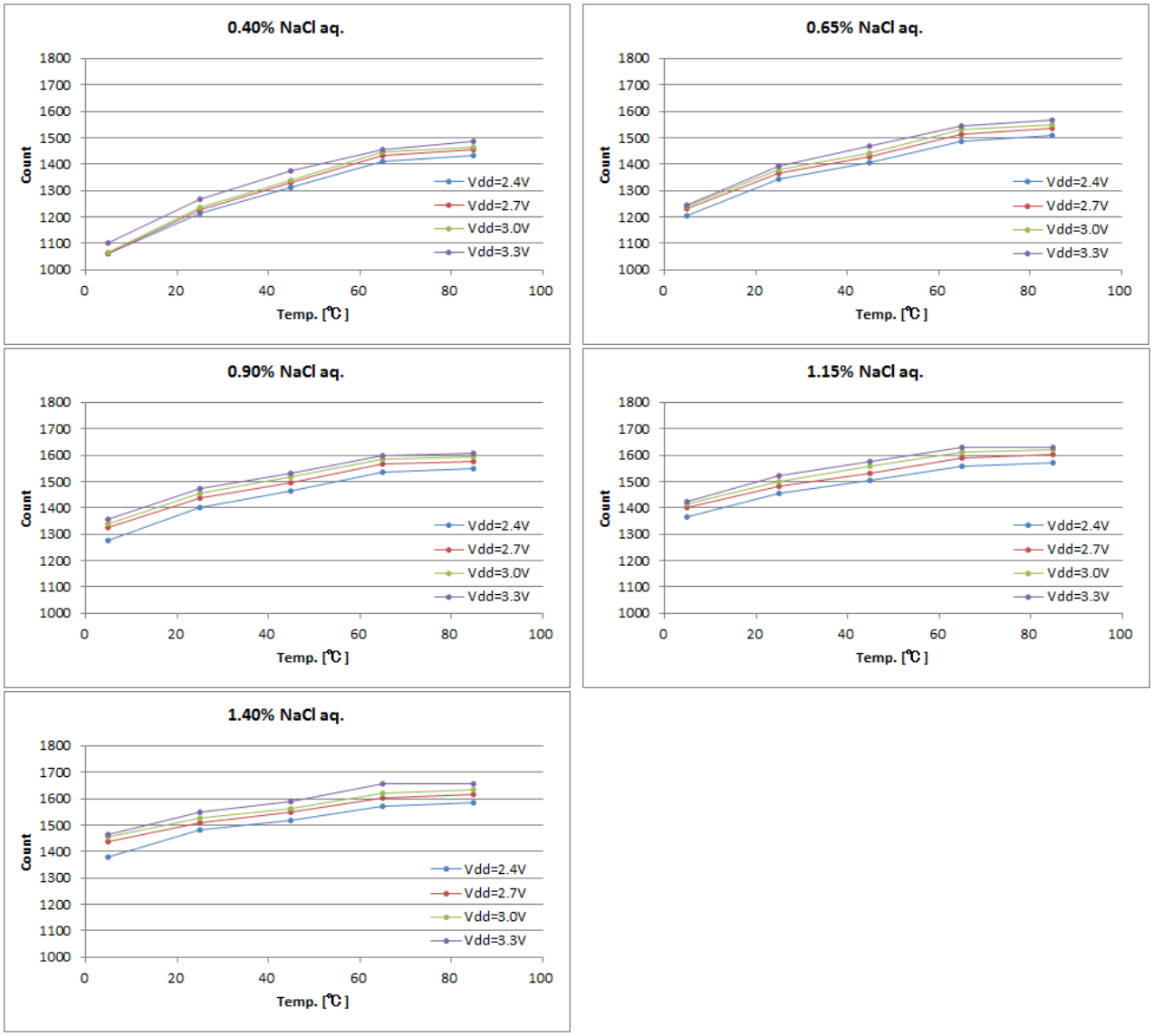
### (4) Measuring dependency of count values on temperature for temperature sensor

Use the thermostatic bath and then confirm the dependency of count values on temperature for temperature sensor. This time, the temperatures were measured in the range between 0°C and 100°C at 5°C intervals. The results were interpolated with logarithmic scale so that the intervals will be 2°C and then the count value was calculated.

The reason why the table data was calculated at smaller intervals in these interpolations are the same as that shown in "APPENDIX A. Creating Example of Thermohygro Table." Since the temperature is used only for correcting salinity concentrations, the interval of salinometer is larger than that of thermohygrometer.

### (5) Measuring dependency of count values on salinity concentration and temperature for salinity concentration sensor

Use the thermostatic bath and then confirm the dependency of count values on salinity concentration and temperature for the salinity concentration sensor. This time, the salinity concentrations were measured in the range between 0.40% and 1.40% at 0.25% intervals and the temperatures were measured in the range between 5°C and 85°C at 20°C intervals. Finally, only the result was used when the power supply voltage Vdd is 3.3V. But, in order to confirm that the impact from power supply voltage cannot be ignored, the measurements were carried out in the range between 2.4V and 3.3V at 0.3V intervals.

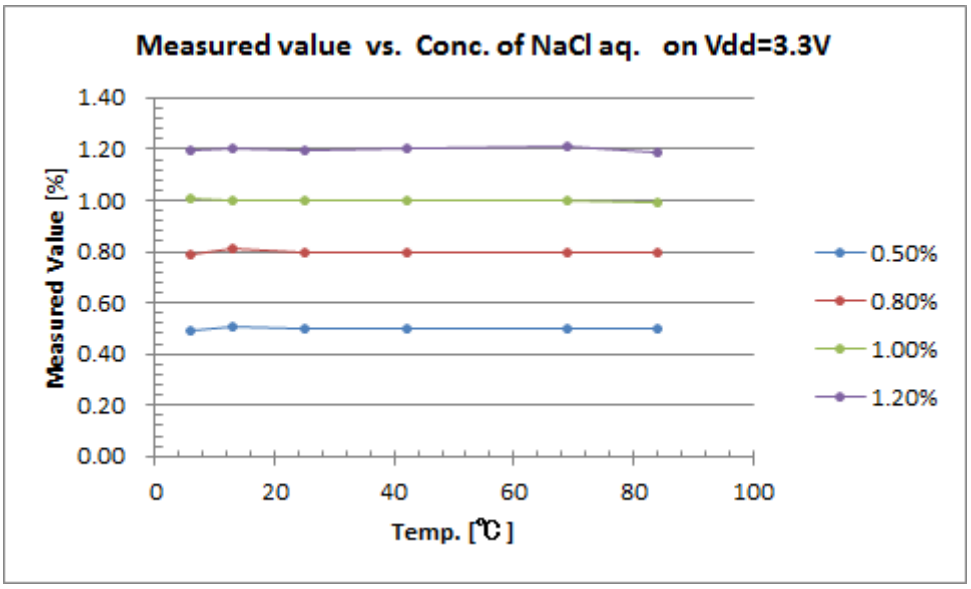



Note that the result above was measured by changing constants defined in main.c in s1c17w22_w23_rfc_temp_hum_analy_gnu17vx as follows:

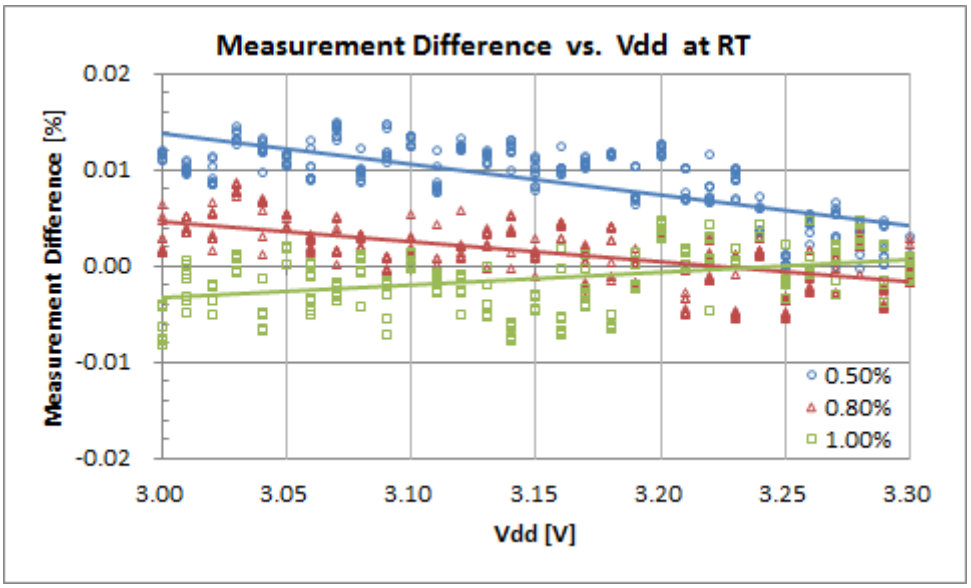
```
HUM_TIME_BASE_COUNT_INDEX       6           // COUNT_1K
```

### (6) Confirming error of salinity concentration measurement result

The salinity concentrations are measured at some points in the range of temperature between 6°C and 84°C and in the range of concentrations between 0.50% and 1.20% by using the parameters obtained above and then the measured salinity concentrations are plotted. Note that the measured salinity concentrations in percentage are the values of tsm.conc1000 divided by 1000.




Also, in order to evaluate the dispersion of measured salinity concentrations during the repeated measurements, as for solutions of three kinds of salinity concentrations, the salinity concentrations were measured ten times in each voltage when changing the power supply voltage Vdd in the range between 3.00V and 3.30V at 0.01V intervals at normal temperature (about 25°C). The graph below shows their dispersion (differences in percentage from prepared concentrations, for example, +0.01% if the measured value of 0.80% prepared concentration is 0.81%).



### (7) Processing time

It is a processing time measured using the [c17 clock] command. It is only for your reference and it does not guarantee the processing time.

Measurement result: salinity concentration: 1.00%, liquid temperature: 24.4°C

| | |
|---|---|
| Measure salinity concentration (runRfcConveringOperation to intRfc0) | 7ms |
| Measure temperature (runRfcConverintgOperation to intRfc1) | 37ms |
| Calculate salinity concentration (running time of getConc function) | 354ms |
| Calculate temperature (running time of getTemp function) | 77ms |
| Display result (running time of dispVal)) | 152ms |

Since the measurements are processed in parallel, the measurement ends when the process which takes longer time ends.

# Revision History

Attachment-1

| Rev. No. | Date | Page | Category | Contents |
|---|---|---|---|---|
| Rev. 1.0 | 2014/8/25 | All | New | |
| Rev. 1.1 | 2016/04/01 | i, 1, 2, 13, 15, 16, 19, 23, 24, 28, 31, 32 | Addition | Descriptions for the contents that were added or modified to support the new operating environment (GNU17 Ver. 3) have been appended. |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

EPSON

# International Sales Operations

## AMERICA

**EPSON ELECTRONICS AMERICA, INC.**
214 Devcon Drive,
San Jose, CA 95112, USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

## EUROPE

**EPSON EUROPE ELECTRONICS GmbH**
Riesstrasse 15, 80992 Munich,
GERMANY
Phone: +49-89-14005-0 FAX: +49-89-14005-110

## ASIA

**EPSON (CHINA) CO., LTD.**
7F, Jinbao Bldg., No.89 Jinbao St.,
Dongcheng District,
Beijing 100005, CHINA
Phone: +86-10-8522-1199 FAX: +86-10-8522-1125

**SHANGHAI BRANCH**
7F, Block B, Hi-Tech Bldg., 900 Yishan Road,
Shanghai 200233, CHINA
Phone: +86-21-5423-5577 FAX: +86-21-5423-4677

**SHENZHEN BRANCH**
12F, Dawning Mansion, Keji South 12th Road,
Hi-Tech Park, Shenzhen 518057, CHINA
Phone: +86-755-2699-3828 FAX: +86-755-2699-3838

**EPSON TAIWAN TECHNOLOGY & TRADING LTD.**
14F, No. 7, Song Ren Road,
Taipei 110, TAIWAN
Phone: +886-2-8786-6688 FAX: +886-2-8786-6660

**EPSON SINGAPORE PTE., LTD.**
1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

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

**SEIKO EPSON CORP.**
**MICRODEVICES OPERATIONS DIVISION**

**Device Sales & Marketing Department**
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5816 FAX: +81-42-587-5117

Document Code: 412808102
First Issue August 2014 Ⓥ
Revised April 2016 in JAPAN