Notice

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

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

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

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

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

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

Application Note

78K0/Lx3

Sample Program (Temperature Measurement)

Temperature Measurement Program Using Port and Timer Functions

This document summarizes the operation of the sample program and describes how to use it. This sample program is used to measure the temperature without using an A/D converter by using a port function (low-level input recognition) and a timer. Specifically, the capacitor is discharged via a thermistor (resistor). The discharge time until the capacitor potential is recognized to be at low level is measured and the thermistor resistance is calculated from the discharge time. The temperature during the measurement is determined based on the data in the provided table showing the resistances and temperature of a thermistor.

Target devices

78K0/LC3 microcontrollers
78K0/LD3 microcontrollers
78K0/LE3 microcontrollers
78K0/LF3 microcontrollers

CONTENTS

Document No. U19542EJ1V0AN00 (1st edition)
Date Published July 2009 N

- **The information in this document is current as of November, 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. In addition, NEC Electronics products are not taken measures to prevent radioactive rays in the product design. When customers use NEC Electronics products with their products, customers shall, on their own responsibility, incorporate sufficient safety measures such as redundancy, fire-containment and anti-failure features to their products in order to avoid risks of the damages to property (including public or social property) or injury (including death) to persons, as the result of defects of NEC Electronics products.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

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

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

  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)

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

(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E0904E

CHAPTER 1 OVERVIEW

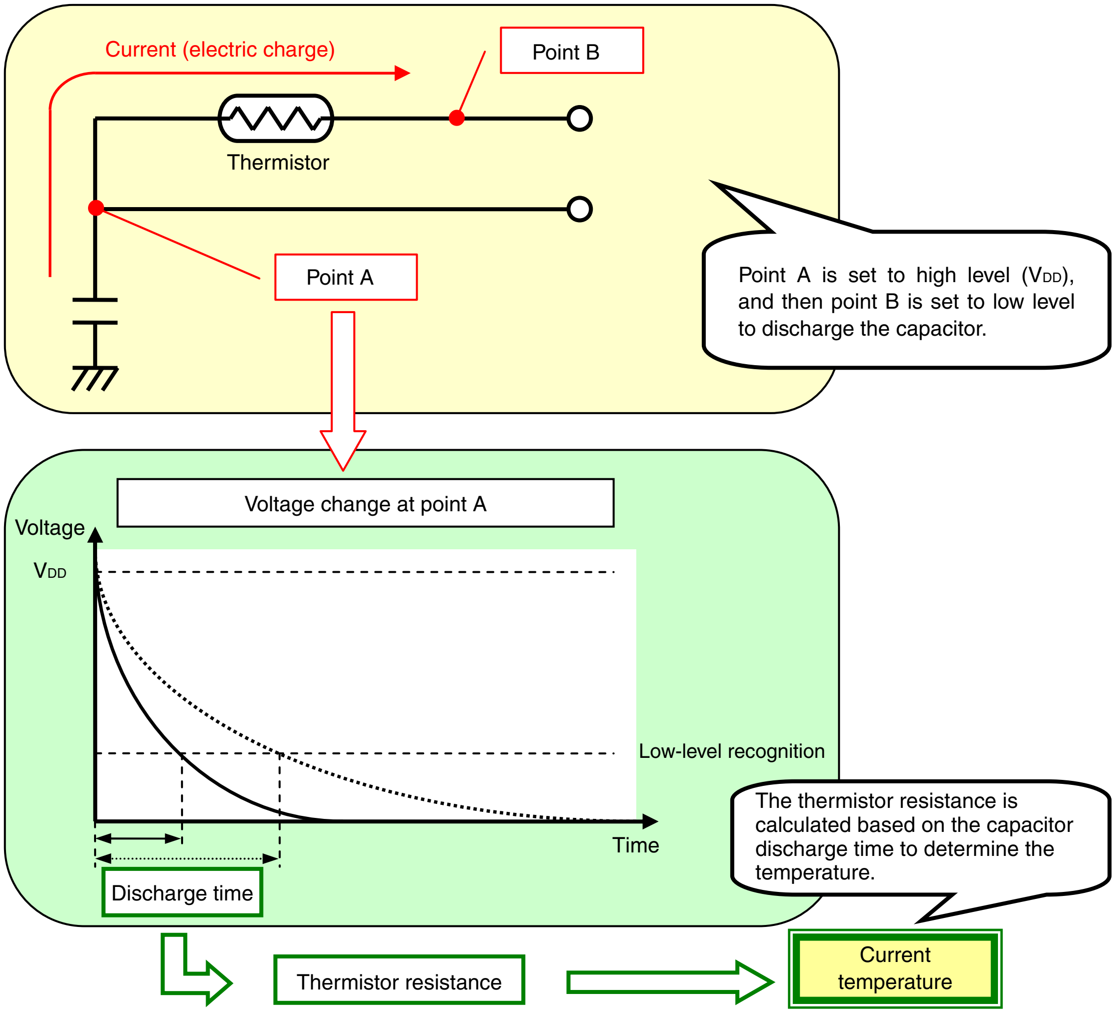
This sample program is used to measure the temperature without using an A/D converter by using a port function (low-level input recognition) and a timer. Specifically, the capacitor is discharged via a thermistor (resistor). The discharge time until the capacitor potential is recognized to be at low level is measured and the thermistor resistance is calculated from the discharge time. The temperature during the measurement is determined based on the data in the provided table showing the resistances and temperature of a thermistor. During the main processing, pulse width measurement processing, temperature acquisition processing, and UART transmission processing are called.

During pulse width measurement processing, the capacitor is discharged and the discharge time is measured by using a fixed resistor for calibration and a thermistor. The discharge time is measured by determining the pulse width by using 16-bit timer/event counter 00.

During temperature acquisition processing, the thermistor resistance is calculated from the measured discharge time of the capacitor and the resistance is converted to a temperature by using the thermistor R-T characteristics specifications. The thermistor resistance is calculated based on the capacitor discharge time and the proportionality of the resistance and capacitor discharge time. The temperature corresponding to the thermistor resistance is obtained from the temperature conversion table corresponding to the thermistor R-T characteristics specifications[Note].

During UART transmission processing, the result of measuring the temperature is converted to ASCII code and transmitted via the serial interface UART6.

Note For details about the temperature conversion table corresponding to the thermistor R-T characteristics specifications, see **2.2 Converting Resistance to Temperature**.



Point A is set to high level (V_{DD}), and then point B is set to low level to discharge the capacitor.

The thermistor resistance is calculated based on the capacitor discharge time to determine the temperature.

(1) Primary initial settings for the peripheral functions

The primary initial settings for the peripheral functions are as follows.

- Disabling interrupts
- Specifying the register bank
- Specifying the stack pointer
- Specifying the ROM and RAM sizes
- Setting up the ports
- Specifying that the CPU clock operate on the internal high-speed oscillation clock (8 MHz)
- Specifying that the peripheral hardware clock operate on the internal high-speed oscillation clock (8 MHz)
- Setting up 16-bit timer/event counter 00 to measure the pulse width
- Specifying 8-bit timer H2 as the base timer (100 ms interval timer) for creating the temperature measurement timing
- Setting up the serial interface UART6 to use for data transmission
- Specifying interrupt masking
- Enabling interrupts

(2) Main processing

During the main processing, discharge time measurement processing, temperature acquisition processing, and UART transmission processing are called.

The processing is called in one-second cycles that are counted by using 8-bit timer H2 as the base timer.

(3) Pulse width measurement processing

The capacitor discharge time is measured by using 16-bit timer/event counter 00 to measure the pulse width.

The capacitor voltage level is detected by the TI000 pin. 16-bit timer/event counter 00 is set up so that it captures the value of 16-bit timer counter 00 at the falling edge of TI000 and generates an interrupt signal (INTTM010).

After the capacitor fully charges, 16-bit timer/event counter 00 is enabled to operate and the capacitor is discharged by using a calibration resistor or thermistor. When the capacitor is fully discharged and TI000 goes to low level, the INTTME010 signal is generated and the discharge time can be obtained from 16-bit timer capture/compare register 010.

(4) Temperature acquisition processing

During temperature acquisition processing, the thermistor resistance is calculated from the measured discharge time of the capacitor and the resistance is converted to a temperature by using the thermistor R-T characteristics specifications.

The thermistor resistance is calculated based on the capacitor discharge time and the proportionality of the resistance and capacitor discharge time. For details, see **2.1 Calculating Thermistor Resistance**.

The thermistor resistance is converted to a temperature by using the temperature conversion table corresponding to the thermistor R-T characteristics specifications. For details, see **2.2 Converting Resistance to Temperature**.

(5) UART transmission processing

During UART transmission processing, the result of measuring the temperature is converted to ASCII code and transmitted via the serial interface UART6.

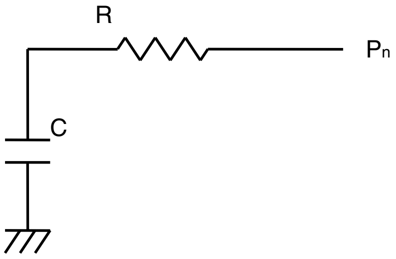
For details about the UART communication settings and the transmitted data, see **4.4 UART Data Transmission Format**.

CHAPTER 2 ALGORITHM FOR MEASURING TEMPERATURE

This chapter describes the temperature-measuring algorithm used in the sample program. The temperature is measured by calculating the thermistor resistance and then converting the resistance to a temperature.

2.1 Calculating Thermistor Resistance

Figure 2-1. Circuit Example



The capacitor is discharged by using a circuit, such as that shown above, and then setting port Pn to low level while the capacitor charges. The time it takes for the capacitor to discharge to 37% of the voltage it was charged to can be calculated by using the following equation.

$$t = R \times C$$

t: Time taken for the capacitor to be discharged by using resistor R [seconds]
R: Resistance of resistor R [Ω]
C: Capacitance of the capacitor [F]

The thermistor resistance is calculated by using the equation above and assuming that the discharge time and resistance are proportional.

If a fixed resistance is used, the capacitor discharge time is constant. The thermistor resistance is calculated by measuring the capacitor discharge time by using a fixed resistor for calibration and a thermistor and then using the following relational equation:

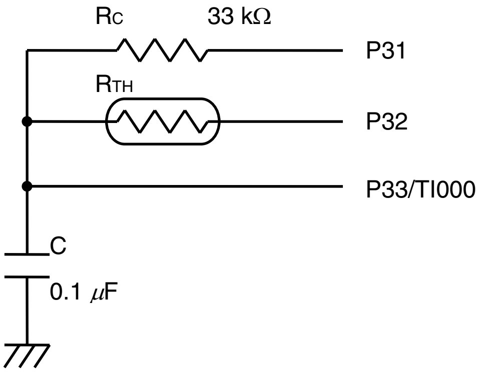
$$t_C : R_C = t_{TH} : R_{TH} \Rightarrow R_{TH} = \frac{R_C \times t_{TH}}{t_C}$$

t_C: Capacitor discharge time if a fixed resistor for calibration is used [seconds]
R_C: Fixed resistor for calibration [Ω]
t_{TH}: Capacitor discharge time if a thermistor is used [seconds]
R_{TH}: Thermistor resistance [Ω]

(1) Measuring the capacitor discharge time

The circuit below is used to measure the capacitor discharge time. The capacitor discharge time is measured by using 16-bit timer/event counter 00 to measure the pulse width and measuring how long TI000 is at high level. For details about the settings for the 16-bit timer/event counter, see **5.1 Initial Settings of Peripheral Functions**.

Figure 2-2. Circuit for Calculating Thermistor Resistance



R_{TH}:	Thermistor
R_C:	Fixed resistor for calibration
P31:	Discharge port used if using a fixed resistor for calibration
P32:	Discharge port used if using a thermistor
P33/TI000:	Port used to detect the completion of charging (P33) and discharging (TI000) the capacitor

Remark The capacitor capacitance and the resistance of the fixed resistor for calibration are determined so that 16-bit timer counter 00 does not overflow when measuring the discharge time.

The procedure for measuring the discharge time is summarized below. For details, see **5.3 Pulse Width Measurement Processing**.

<1> Set P33 to high-level output and charge the capacitor.[Note 1]

<2> Set P33 as an input port so that it can be used as the TI000 pin.

<3> Set the port to be used to discharge the capacitor (P31 or P32)[Note 2] to low-level output and start discharging the capacitor.

<4> At the same time, enable 16-bit timer/event counter 00 and start measuring the discharge time.

<5> Wait until the capacitor fully discharges. If necessary, count how many times 16-bit timer counter 00 overflows.

<6> Obtain the measured capacitor discharge time from 16-bit timer capture/compare register 010.

<7> Disable 16-bit timer/event counter 00 and set the port used to discharge the capacitor (P31 or P32)[Note 2] as an input port.

Notes 1. Assuming that the CMOS output resistance of the microcontroller during high-level output is 2 kΩ, the time constant τ (the time required for charging the capacitor up to 63% of the power supply voltage (V_{DD})) can be calculated by using the following equation.

$\tau = 0.1\ [\mu F] \times 2\ [k\Omega] = 0.2\ [ms]$

In the sample program, the charge time is specified as 5τ = 2 ms so that the capacitor is sufficiently charged.

2. P31 is used if a fixed resistor for calibration is used to discharge the capacitor. P32 is used if using a thermistor.

(2) Calculating the thermistor resistance

The thermistor resistance is calculated by using the measured capacitor discharge time and the equation below. The capacitor discharge time used in the calculation is a 16-bit value read from 16-bit timer capture/compare register 010. The capacitor discharge time if a thermistor is used is calculated by using a value of up to 17 bits[Note 1], including the number of overflows that occurred.

$$R_{TH} = \frac{R_C \times (CNT_{TH} + \text{number of overflows} \times 10000H)}{CNT_C}$$

R_{TH}:	Thermistor resistance [100 Ω]
R_C:	Fixed resistor for calibration [100 Ω]
CNT_{TH}:	Capacitor discharge time if a thermistor is used[Note 2]
CNT_C:	Capacitor discharge time if a fixed resistor for calibration is used[Note 2]
Number of overflows:	Number of times 16-bit timer counter 00 overflowed[Note 2] when the capacitor discharge time if a thermistor is used is measured

Notes 1. The resistance measurement range with respect to the temperature measurement range (42.0 to 32.0°C) in the sample program is 24.5 to 37.0 kΩ. If the capacitor discharge time is calculated by using the equation below when the number of overflows is at least 2, the value will be outside the resistance measurement range. The thermistor resistance will not be calculated, and the calculation will be processed as a measurement error when the number of overflows is at least 2.

2. The capacitor discharge pulse width is measured by using the peripheral hardware clock (f_{PRS}) as the count clock. Therefore, the discharge time of the capacitor used to calculate the thermistor resistance (CNT_C and CNT_{TH}) and the capacitor discharge time (t_C and t_{TH} [seconds]) can be calculated by using the following equations.

$$t_C \text{ [seconds]} = \frac{CNT_C}{f_{PRS}}$$

$$t_{TH} \text{ [seconds]} = \frac{CNT_{TH} + \text{number of overflows} \times 10000H}{f_{PRS}}$$

t_C:	Capacitor discharge time if a fixed resistor for calibration is used [seconds]
t_{TH}:	Capacitor discharge time if a thermistor is used [seconds]
f_{PRS}:	Peripheral hardware clock frequency [Hz]

2.2 Converting Resistance to Temperature

The thermistor resistance is converted to a temperature by obtaining the temperature corresponding to the thermistor resistance from the temperature conversion table. The temperature conversion table corresponds to the thermistor R-T characteristics specifications.

The thermistor 503ET, made by Ishizuka Electronics Corporation, is used in the sample program. The thermistor R-T characteristics specifications for the 503ET are shown in the following table.

Table 2-1. Thermistor R-T Characteristics Specifications (32.0 to 42.0°C)

Temperature [°C]	Maximum Resistance [kΩ]	Standard Resistance [kΩ]	Minimum Resistance [kΩ]	Allowable Temperature Error [°C]
31	39.82	38.56	37.30	−0.8 to +0.8
32	38.18	36.95	35.74	−0.8 to +0.8
33	36.62	35.43	34.25	−0.8 to +0.9
34	35.13	33.98	33.98	−0.8 to +0.9
35	33.71	32.59	32.83	−0.9 to +0.9
36	32.36	31.27	31.27	−0.9 to +0.9
37	31.07	30.01	30.01	−0.9 to +0.9
38	29.84	28.81	28.81	−0.9 to +0.9
39	28.67	27.67	27.67	−0.9 to +1.0
40	27.55	26.58	26.58	−0.9 to +1.0
41	26.46	25.52	25.52	−0.9 to +1.0
42	25.43	24.51	24.51	−1.0 to +1.0

Remark The data in the temperature measurement range (32.0 to 42.0°C) and the data calculated when creating the temperature conversion table are taken from the thermistor R-T characteristics specifications in the table above.

The temperature conversion table (Table 2-2) is created based on the temperatures and standard resistances in Table 2-1. These standard resistances are corrected in 100 Ω units so that the graph based on the data in Table 2-1 is a straight line and the temperature with respect to the resistance is calculated in 0.1°C units. The temperature with respect to the resistance is calculated by using the equation below. The equation assumes that the resistance and temperature of the thermistor are proportional if the thermistor temperature changes by 1°C.

$$T_{TH} = \underbrace{T_0 - \frac{\overbrace{R_{TH} - R_0}^{\text{<2>}}}{\underbrace{R_0 - R_1}_{\text{<1>}}}}_{\text{<3>}}$$

T_{TH}: Thermistor temperature [°C]

T_0: Reference temperature [°C][Note]

R_{TH}: Thermistor resistance [100 Ω]

R_0: Standard resistance at the reference temperature [100 Ω][Note]

R_1: Standard resistance at 1°C below the reference temperature [100 Ω][Note]

<1> The change in the resistance per degree in the range that indicates the thermistor resistance is calculated.

<2> How much the thermistor resistance has changed from the resistance at the reference temperature is calculated.

<3> The thermistor temperature is calculated by calculating how much the temperature has changed from the reference temperature with respect to the thermistor resistance based on <1> and <2> and then subtracting that amount from the reference temperature.

Note Specify these so that $R_1 < R_{TH} < R_0$. For example, if R_{TH} = 25 kΩ, T_0 = 42°C, R_0 = 24.51 kΩ, and R_1 = 25.52 kΩ.

Table 2-2. Temperature Conversion Table

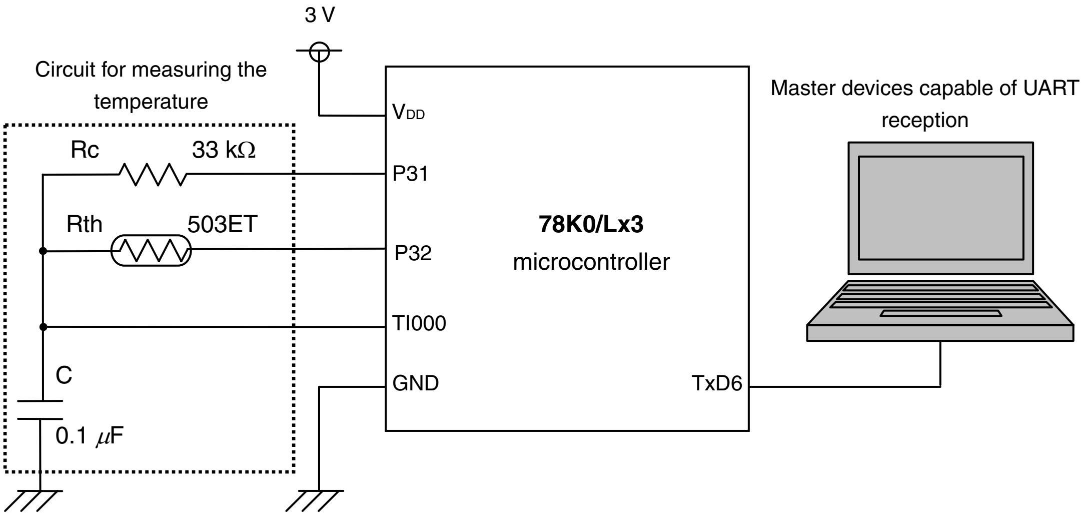
Resistance [100 Ω]	Temperature [°C]
245	42.0
246	41.9
247	41.8
248	41.7
249	41.6
250	41.5
251	41.4
252	41.3
253	41.2
254	41.1
255	41.0
256	40.9
257	40.8
258	40.7
259	40.6
260	40.5
261	40.5
262	40.4
263	40.3
264	40.2
265	40.1
266	40.0
267	39.9
268	39.8
269	39.7
270	39.6
271	39.5
272	39.4
273	39.3
274	39.2
275	39.2
276	39.1
277	39.0
278	38.9
279	38.8
280	38.7
281	38.6
282	38.5
283	38.4
284	38.4
285	38.3
286	38.2
287	38.1
288	38.0
289	37.9
290	37.8
291	37.8
292	37.7
293	37.6
294	37.5
295	37.4
296	37.3
297	37.3
298	37.2
299	37.1
300	37.0
301	36.9
302	36.8
303	36.8
304	36.7
305	36.6
306	36.5
307	36.5
308	36.4
309	36.3
310	36.2
311	36.1
312	36.1
313	36.0
314	35.9
315	35.8
316	35.8
317	35.7
318	35.6
319	35.5
320	35.4
321	35.4
322	35.3
323	35.2
324	35.1
325	35.1
326	35.0
327	34.9
328	34.8
329	34.8
330	34.7
331	34.6
332	34.6
333	34.5
334	34.4
335	34.3
336	34.3
337	34.2
338	34.1
339	34.1
340	34.0
341	33.9
342	33.8
343	33.8
344	33.7
345	33.6
346	33.6
347	33.5
348	33.4
349	33.4
350	33.3
351	33.2
352	33.2
353	33.1
354	33.0
355	33.0
356	32.9
357	32.8
358	32.8
359	32.7
360	32.6
361	32.6
362	32.5
363	32.4
364	32.4
365	32.3
366	32.2
367	32.2
368	32.1
369	32.0
370	32.0

CHAPTER 3 CIRCUIT DIAGRAM

This chapter shows a diagram of the circuit used in the sample program and describes the peripheral hardware.

3.1 Circuit Diagram

The circuit diagram is shown below.


3 V
Circuit for measuring the temperature
Rc
33 kΩ
Rth
503ET
C
0.1 μF
VDD
P31
P32
TI000
GND
TxD6
78K0/Lx3
microcontroller
Master devices capable of UART reception

Cautions 1. Connect the AV$_{REF}$ pin directly to V$_{DD}$ (3 V supply).

2. Connect the AV$_{SS}$ pin directly to GND.

3. Leave all pins in the circuit diagram and all unused pin except the AV$_{REF}$ and AV$_{SS}$ pins open (unconnected), because they are used as output ports.

4. Connect the TxD6 pin to a device capable of UART reception.

3.2 Peripheral Hardware

The peripheral hardware is described below.

(1) UART communication device (TxD6)

Connect a device to use for UART reception to the TxD6 pin.

(2) Circuit for measuring the temperature (TI000, P31, P32)

Connect a 0.1 μF capacitor to the TI000 pin, a 33 kΩ fixed resistor for calibration to the P31 pin, and a thermistor to the P32 pin.

The high-sensitivity thermistor 503ET, made by Ishizuka Electronics Corporation, is used in the sample program.

Cautions 1. Connect the AV$_{REF}$ pin directly to V$_{DD}$ (3 V supply).

2. Connect the AV$_{SS}$ pin directly to GND.

CHAPTER 4 SOFTWARE

This chapter describes the files in the compressed file to be downloaded, the internal peripheral functions of the microcontroller, the initial settings, and the UART data transmission format. This chapter also provides an operational overview of the sample program and shows flowcharts.

4.1 Included Files

The compressed file to be downloaded includes the following files.

File Name	Description	Included Compressed Files (*.zip)	
		ZIP	PM 32
main.asm (assembly language version) main.c (C language version)	Source files for hardware initialization processing of the microcontroller, main processing, pulse width measurement processing, temperature acquisition processing, and UART transmission processing	●[Note]	●[Note]
op.asm	Assembler source file for setting up the option byte (This file is used to set up the watchdog timer and internal low-speed oscillator.)	●	●
78K0Lx3_Thermistor.prw	Workspace file for the integrated development environment PM plus		●
78K0Lx3_Thermistor.prj	Project file for the integrated development environment PM plus		●

Note The assembly language version includes main.asm and the C language version includes main.c.

Remark ZIP : Includes only source files.

PM 32 : Includes files used for the integrated development environment PM plus.

4.2 Used Internal Peripheral Functions

The following peripheral functions provided in the microcontroller are used in the sample program:

- Internal high-speed oscillator
  This oscillator is used for the CPU clock and peripheral hardware clock.

- 8-bit timer H2
  This timer is used as a 100 ms interval timer to create the timing for measuring the temperature.

- 16-bit timer/event counter 00
  This timer/event counter is used to measure the discharge pulse width of the capacitor.

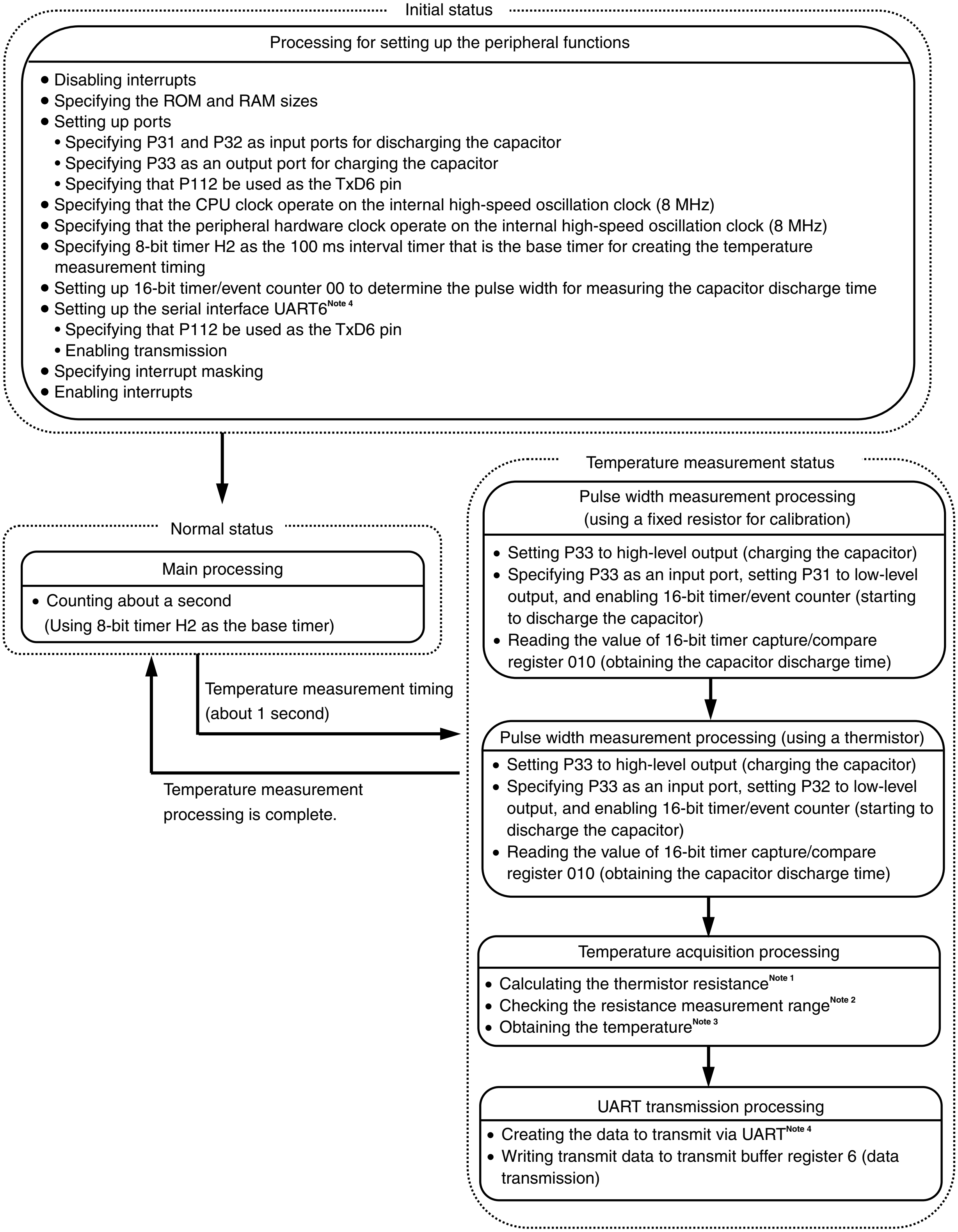
- Serial interface UART6
  This interface is used to transmit the result of measuring the temperature.

4.3 Initial Settings and Operational Overview

In the sample program, the ports, 8-bit timer H2, 16-bit timer/event counter 00, and the serial interface UART6 are set up and the clock frequency is selected as part of the initial settings for the peripheral functions. After the initial settings for the peripheral functions are set up, the capacitor discharge time is measured by using a fixed resistor for calibration and a thermistor, the temperature is obtained from the capacitor discharge time, and the result of measuring the temperature is transmitted by using UART about once a second. These seconds are counted by using 8-bit timer H2 as the base timer.

For details, see the status transition diagram below.

Note For details about the UART communication settings and the transmitted data, see **4.4 UART Data Transmission Format**.



Initial status

Processing for setting up the peripheral functions

- Disabling interrupts
- Specifying the ROM and RAM sizes
- Setting up ports
  - Specifying P31 and P32 as input ports for discharging the capacitor
  - Specifying P33 as an output port for charging the capacitor
  - Specifying that P112 be used as the TxD6 pin
- Specifying that the CPU clock operate on the internal high-speed oscillation clock (8 MHz)
- Specifying that the peripheral hardware clock operate on the internal high-speed oscillation clock (8 MHz)
- Specifying 8-bit timer H2 as the 100 ms interval timer that is the base timer for creating the temperature measurement timing
- Setting up 16-bit timer/event counter 00 to determine the pulse width for measuring the capacitor discharge time
- Setting up the serial interface UART6[Note 4]
  - Specifying that P112 be used as the TxD6 pin
  - Enabling transmission
- Specifying interrupt masking
- Enabling interrupts

Normal status

Main processing

- Counting about a second (Using 8-bit timer H2 as the base timer)

Temperature measurement timing (about 1 second)

Temperature measurement processing is complete.

Temperature measurement status

Pulse width measurement processing (using a fixed resistor for calibration)

- Setting P33 to high-level output (charging the capacitor)
- Specifying P33 as an input port, setting P31 to low-level output, and enabling 16-bit timer/event counter (starting to discharge the capacitor)
- Reading the value of 16-bit timer capture/compare register 010 (obtaining the capacitor discharge time)

Pulse width measurement processing (using a thermistor)

- Setting P33 to high-level output (charging the capacitor)
- Specifying P33 as an input port, setting P32 to low-level output, and enabling 16-bit timer/event counter (starting to discharge the capacitor)
- Reading the value of 16-bit timer capture/compare register 010 (obtaining the capacitor discharge time)

Temperature acquisition processing

- Calculating the thermistor resistance[Note 1]
- Checking the resistance measurement range[Note 2]
- Obtaining the temperature[Note 3]

UART transmission processing

- Creating the data to transmit via UART[Note 4]
- Writing transmit data to transmit buffer register 6 (data transmission)

Notes 1. For details about the processing, see **2.1 Calculating Thermistor Resistance**.

2. The resistance measurement range with respect to the temperature measurement range (42.0 to 32.0°C) in the sample program is 24.5 to 37.0 kΩ.

3. For details about the processing, see **2.2 Converting Resistance to Temperature**.

4. For details about the UART communication settings and the transmitted data, see **4.4 UART Data Transmission Format**.

4.4 UART Data Transmission Format

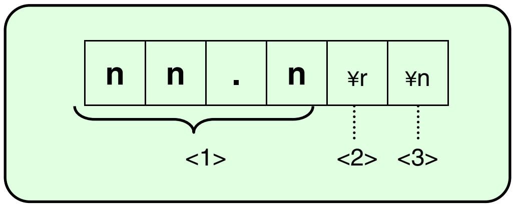
The data to transmit via the serial interface UART6 is described below.

The following table shows the settings for the serial interface UART6.

Item to Specify	Setting
Baud rate	115,200 bps
Character length of transmit data	8 bits
Parity bit	Not output
Number of stop bits	1
Start bit	LSB

Data is transmitted about every second. 6 bytes of data is transmitted per transmission. The result of measuring the temperature is converted to ASCII code and then transmitted. Figure 4-1 shows the data transmission format.

Figure 4-1. UART Data Transmission Format



<1> The measured temperature is a decimal value. The temperature is measured in the range from 32.0 to 42.0°C. If the result of measuring the temperature is an error, the data to transmit is represented as “**.*” (n = 0 to 9, *).

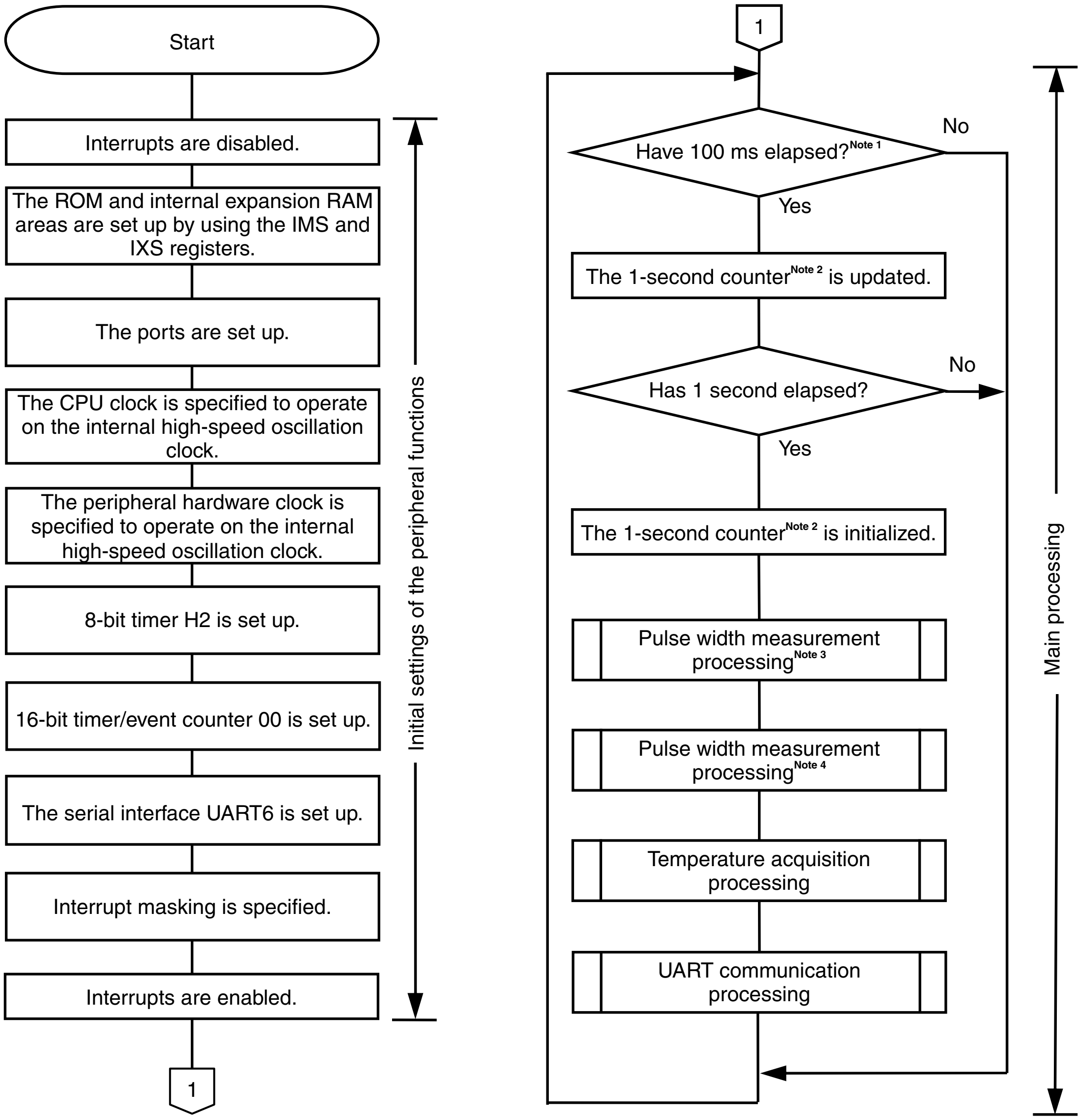
<2> Return code

<3> Line feed code

4.5 Flowcharts

The flowcharts for the sample program are shown below.

<Processing for setting up the peripheral functions>


Start
Interrupts are disabled.
The ROM and internal expansion RAM areas are set up by using the IMS and IXS registers.
The ports are set up.
The CPU clock is specified to operate on the internal high-speed oscillation clock.
The peripheral hardware clock is specified to operate on the internal high-speed oscillation clock.
8-bit timer H2 is set up.
16-bit timer/event counter 00 is set up.
The serial interface UART6 is set up.
Interrupt masking is specified.
Interrupts are enabled.
1
Initial settings of the peripheral functions
1
Have 100 ms elapsed?Note 1
No
Yes
The 1-second counterNote 2 is updated.
Has 1 second elapsed?
No
Yes
The 1-second counterNote 2 is initialized.
Pulse width measurement processingNote 3
Pulse width measurement processingNote 4
Temperature acquisition processing
UART communication processing
Main processing

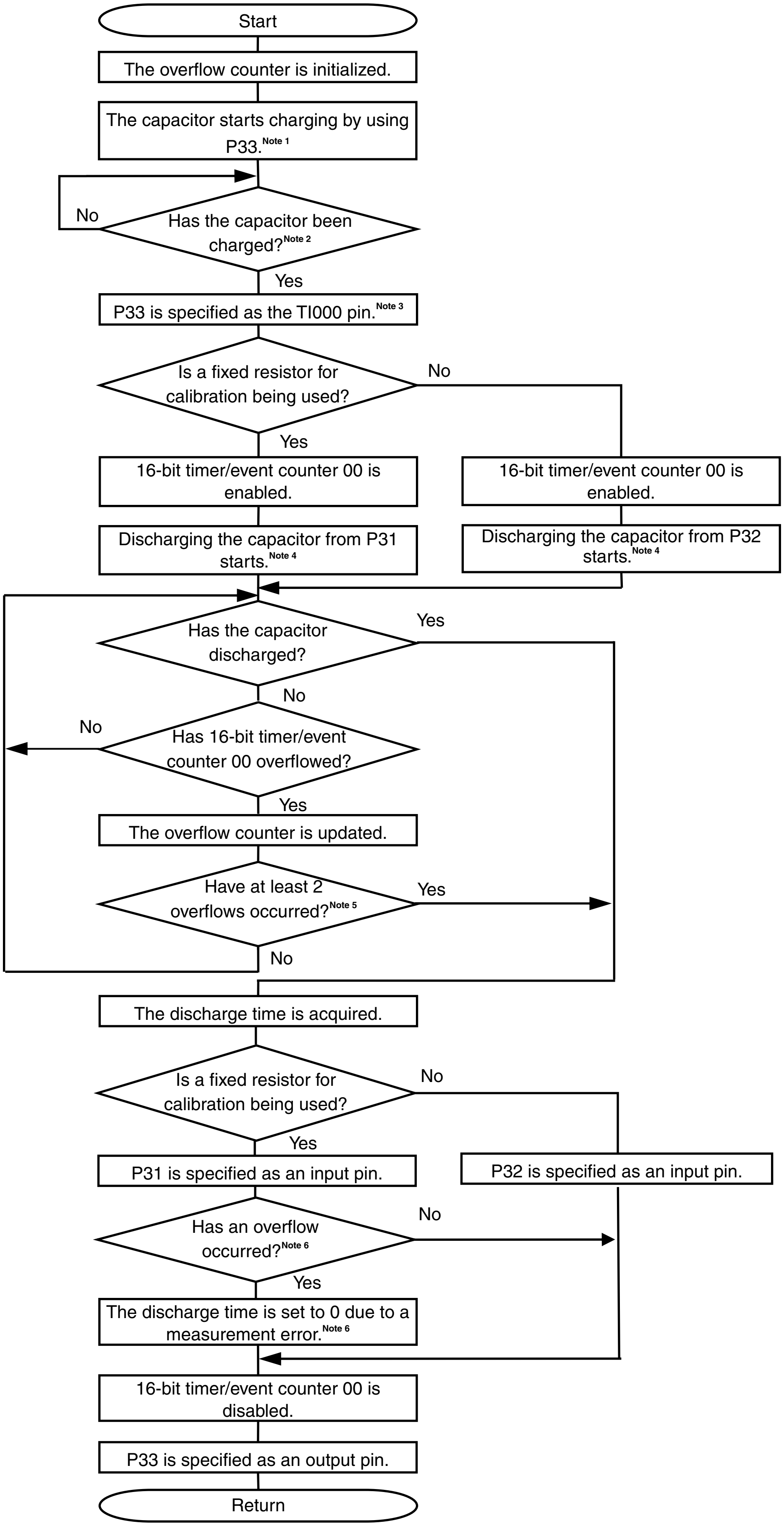
Notes 1. 8-bit timer H2 is used as the base timer.

2. This variable counts about one second by using 8-bit timer H2 as the base timer.

3. The capacitor discharge pulse width is measured by using a fixed resistor for calibration.

4. The capacitor discharge pulse width is measured by using a thermistor.

<Pulse width measurement processing>


Start
The overflow counter is initialized.
The capacitor starts charging by using P33.Note 1
Has the capacitor been charged?Note 2
No
Yes
P33 is specified as the TI000 pin.Note 3
Is a fixed resistor for calibration being used?
No
Yes
16-bit timer/event counter 00 is enabled.
Discharging the capacitor from P31 starts.Note 4
16-bit timer/event counter 00 is enabled.
Discharging the capacitor from P32 starts.Note 4
Has the capacitor discharged?
Yes
No
Has 16-bit timer/event counter 00 overflowed?
No
Yes
The overflow counter is updated.
Have at least 2 overflows occurred?Note 5
Yes
No
The discharge time is acquired.
Is a fixed resistor for calibration being used?
No
Yes
P31 is specified as an input pin.
P32 is specified as an input pin.
Has an overflow occurred?Note 6
No
Yes
The discharge time is set to 0 due to a measurement error.Note 6
16-bit timer/event counter 00 is disabled.
P33 is specified as an output pin.
Return

Notes 1. P33 is set to high-level output.

2. The system waits about 2 ms for the capacitor to charge.

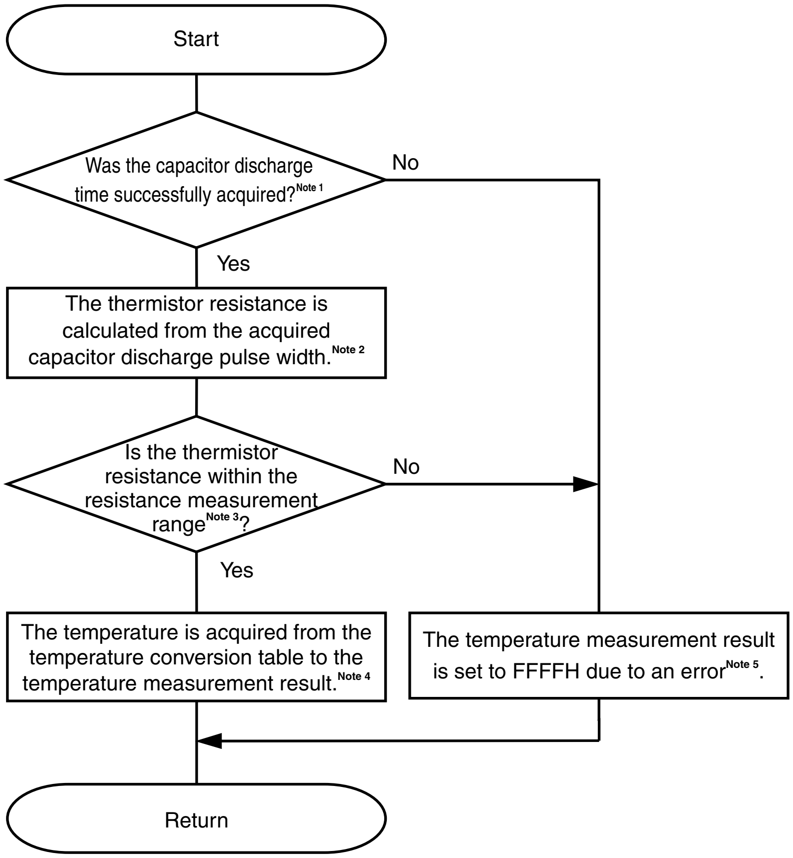
3. P33 is specified as an input port.

4. P31 or P32 is set to low-level output.

5. The fixed resistor for calibration and capacitor, which are used to measure the discharge time by using a fixed resistor for calibration, are selected such that 16-bit timer counter 00 will not overflow. If at least two overflows occur when the discharge time is measured by using a thermistor, the result of calculating the thermistor resistance will be outside the measurement range. Therefore, the occurrence of at least two overflows is considered as a measurement error and measuring the capacitor discharge time is suspended.

6. The fixed resistor for calibration and capacitor, which are used to measure the discharge time by using a fixed resistor for calibration, are selected such that 16-bit timer counter 00 will not overflow. Therefore, if an overflow occurs when measuring the discharge pulse width for calibration, the discharge time is set to 0 due to a measurement error.

<Temperature acquisition processing>


Start
Was the capacitor discharge time successfully acquired?Note 1
No
Yes
The thermistor resistance is calculated from the acquired capacitor discharge pulse width.Note 2
Is the thermistor resistance within the resistance measurement rangeNote 3?
No
Yes
The temperature is acquired from the temperature conversion table to the temperature measurement result.Note 4
The temperature measurement result is set to FFFFH due to an errorNote 5.
Return

Notes 1. Whether the capacitor discharge time was successfully acquired when a fixed resistor for calibration was used or whether at least two overflows occurred when the capacitor discharge time was measured by using a thermistor is determined.

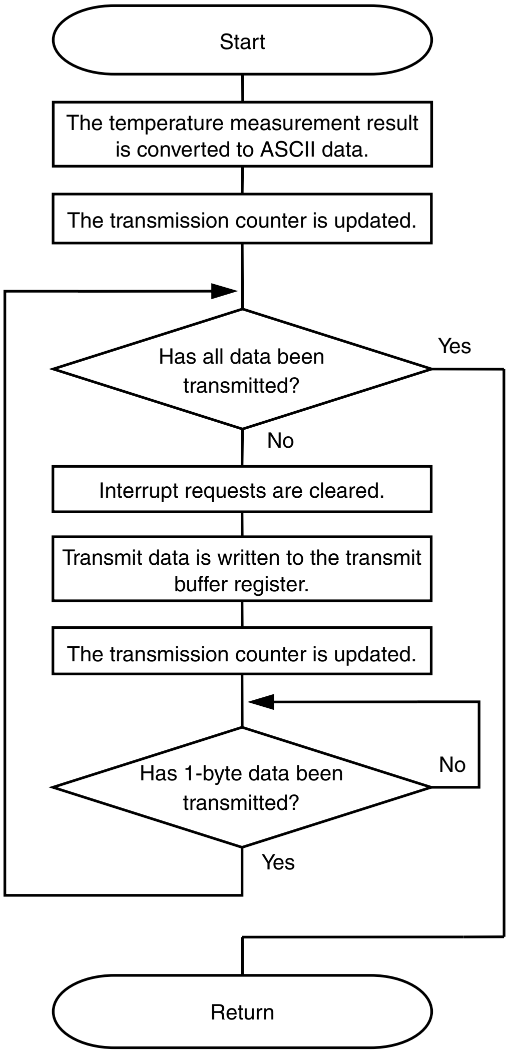
2. The thermistor resistance is calculated by using the ratio between the discharge time and resistance. For details, see **2.1 Calculating Thermistor Resistance**.

3. The resistance measurement range with respect to the temperature measurement range (32.0 to 42.0°C) is 37.0 to 24.5 kΩ.

4. The temperature is acquired from the temperature conversion table. For details, see **2.2 Converting Resistance to Temperature**.

5. The result is specified as FFFFH.

<UART transmission processing>


Start
The temperature measurement result is converted to ASCII data.
The transmission counter is updated.
Has all data been transmitted?
Yes
No
Interrupt requests are cleared.
Transmit data is written to the transmit buffer register.
The transmission counter is updated.
Has 1-byte data been transmitted?
No
Yes
Return

CHAPTER 5 SETUP

This chapter describes the initial settings for the peripheral functions and the processing of the 78K0/LF3 sample program.

For details about how to set up the option byte, vector table, memory space, stack pointer, and registers, as well as how to specify the clock frequency, see the user's manual and sample program for each product (78K0/Lx3).

For details about assembler instructions, see the **78K0 Microcontroller Instructions User's Manual**.

5.1 Initial Settings of Peripheral Functions

(1) Variable definitions

The following variables are defined in assembly language:

<1> R1SECCNT: This 1-second counter counts by using 8-bit timer H2 as the base timer.

<2> ROVFCNT: This counter counts the number of overflows when the capacitor discharge time is measured.

<3> RTXBUF: The data to transmit by using UART communication is stored in this array. The result of measuring the temperature is converted to ASCII code and then stored in this array.

<4> RCALBCNT: The capacitor discharge time measured by using a fixed resistor for calibration is stored in this variable.

<5> RTHERMCNT: The capacitor discharge time measured by using a thermistor is stored in this variable.

<6> RHEAT: The result of measuring the temperature is stored in this variable.

<7> RTEMP16A: This area is used for 16-bit operations.

<8> RTEMP16B: This area is used for 16-bit operations.

<9> RTEMP32: This area is used for 32-bit operations.

```
         ;===================================================================
         ;
         ;       RAM definitions
         ;
         ;===================================================================
         DTHERMO      DSEG    SADDR
<1>----- R1SECCNT:    DS      1           ;Counts the time by using 100 ms (TMH2) as the base
         timer.
          C1SEC       EQU     (1000/100)  ;Used to count 1 second.

<2>----- ROVFCNT:     DS      1           ;Counter that counts the number of times TM00
         overflows

<3>----- RTXBUF:      DS      6           ;Transmit data buffer

         DTHERMOP     DSEG    SADDRP      ;RAM related to measuring the temperature
<4>----- RCALBCNT:    DS      2           ;Used to acquire the value for measuring the TI000
         pulse width for calibration.
<5>----- RTHERMCNT:   DS      2           ;Used to acquire the value for measuring the TI000
         pulse width for the thermistor.
<6>----- RHEAT:       DS      2           ;Saves the calculated temperature.  * FFFFH is
         output for a measurement error.

<7>----- RTEMP16A:    DS      2           ;Variable used to calculate the resistance
<8>----- RTEMP16B:    DS      2           ;Variable used to calculate the resistance
<9>----- RTEMP32:     DS      4           ;Variable used to calculate the resistance
```

The following variables are defined in C language:

<1> uc1secCnt: This 1-second counter counts by using 8-bit timer H2 as the base timer.

<2> ushCalibrationCnt: The capacitor discharge time measured by using a fixed resistor for calibration is stored in this variable.

<3> ushThermistorCnt: The capacitor discharge time measured by using a thermistor is stored in this variable.

<4> ucOVFcnt: This counter counts the number of overflows when the capacitor discharge time is measured.

<5> ushHeatData: The result of measuring the temperature is stored in this variable.

<6> ucTxBuffer[6]: The data to transmit by using UART communication is stored in this array. The result of measuring the temperature is converted to ASCII code and then stored in this array.

<7> ucTxBufferCounter: This counter counts the number of data units transmitted during UART transmission processing.

```
          /*=================================================================

                  RAM definitions

          =================================================================*/
<1>-----  unsigned char uc1secCnt;           /* Counts 1 second by using 100 ms (TMH2) as the
          base timer. */
          #define         TMH2_1SEC       (1000/100)      /* Used to count 1 second. */

<2>-----  unsigned short ushCalibrationCnt;  /* Used to acquire the value for measuring the
          TI000 pulse width for calibration. */
<3>-----  unsigned short ushThermistorCnt;   /* Used to acquire the value for measuring the
          TI000 pulse width for the thermistor. */
<4>-----  unsigned char ucOVFcnt;            /* Counter that counts the number of times TM00
          overflows */
<5>-----  unsigned short ushHeatData;        /* Saves the calculated temperature.  * FFFFH is
          output for a measurement error. */

<6>-----  unsigned char ucTxBuffer[6];       /* Transmit data buffer */
<7>-----  unsigned char ucTxBufferCounter;   /* Transmission counter */
```

(2) Table definitions

The temperature conversion table used to convert the resistance to a temperature in assembly language is defined as shown below. If the thermistor resistance is within the measurement range, the temperature (BCD) is acquired by calculating the offset of the thermistor resistance from the minimum resistance (24.5 kΩ) in the measurement range and then converting that offset to the offset from the start address in the temperature conversion table. For details about the data in the temperature conversion table, see **2.2 Converting Resistance to Temperature**.

```
;==================================================================
;
;       ROM definitions
;
;==================================================================
CREGACC CSEG    UNITP
;------------------------------------------------------------------
;       Table used to convert the resistance to a temperature
;------------------------------------------------------------------
;       The temperature is referenced according to the offset based on 24.5 kΩ [100 Ω].
;       BCD [0.1°C] is referenced.
;------------------------------------------------------------------
TR2HEAT:
        DW      0420H           ;24.5 kΩ → 42.0
        DW      0419H           ;24.6 kΩ → 41.9
        DW      0418H           ;24.7 kΩ → 41.8
        DW      0417H           ;24.8 kΩ → 41.7
        DW      0416H           ;24.9 kΩ → 41.6
        DW      0415H           ;25.0 kΩ → 41.5
        DW      0414H           ;25.1 kΩ → 41.4
        DW      0413H           ;25.2 kΩ → 41.3
        DW      0412H           ;25.3 kΩ → 41.2
        DW      0411H           ;25.4 kΩ → 41.1
        DW      0410H           ;25.5 kΩ → 41.0
        DW      0409H           ;25.6 kΩ → 40.9
        DW      0408H           ;25.7 kΩ → 40.8
        DW      0407H           ;25.8 kΩ → 40.7
        DW      0406H           ;25.9 kΩ → 40.6
        DW      0405H           ;26.0 kΩ → 40.5
        DW      0405H           ;26.1 kΩ → 40.5
        DW      0404H           ;26.2 kΩ → 40.4
        DW      0403H           ;26.3 kΩ → 40.3
        DW      0402H           ;26.4 kΩ → 40.2
        DW      0401H           ;26.5 kΩ → 40.1
        DW      0400H           ;26.6 kΩ → 40.0
        DW      0399H           ;26.7 kΩ → 39.9
        DW      0398H           ;26.8 kΩ → 39.8
        DW      0397H           ;26.9 kΩ → 39.7
        DW      0396H           ;27.0 kΩ → 39.6
        DW      0395H           ;27.1 kΩ → 39.5
        DW      0394H           ;27.2 kΩ → 39.4
        DW      0393H           ;27.3 kΩ → 39.3
        DW      0392H           ;27.4 kΩ → 39.2
        DW      0392H           ;27.5 kΩ → 39.2
        DW      0391H           ;27.6 kΩ → 39.1
        DW      0390H           ;27.7 kΩ → 39.0
        DW      0389H           ;27.8 kΩ → 38.9
        DW      0388H           ;27.9 kΩ → 38.8
        DW      0387H           ;28.0 kΩ → 38.7
        DW      0386H           ;28.1 kΩ → 38.6
        DW      0385H           ;28.2 kΩ → 38.5
        DW      0384H           ;28.3 kΩ → 38.4
        DW      0384H           ;28.4 kΩ → 38.4
        DW      0383H           ;28.5 kΩ → 38.3
        DW      0382H           ;28.6 kΩ → 38.2
        DW      0381H           ;28.7 kΩ → 38.1
        DW      0380H           ;28.8 kΩ → 38.0
        DW      0379H           ;28.9 kΩ → 37.9
        DW      0378H           ;29.0 kΩ → 37.8
        DW      0378H           ;29.1 kΩ → 37.8
        DW      0377H           ;29.2 kΩ → 37.7
        DW      0376H           ;29.3 kΩ → 37.6
        DW      0375H           ;29.4 kΩ → 37.5
        DW      0374H           ;29.5 kΩ → 37.4
        DW      0373H           ;29.6 kΩ → 37.3
        DW      0373H           ;29.7 kΩ → 37.3
        DW      0372H           ;29.8 kΩ → 37.2
        DW      0371H           ;29.9 kΩ → 37.1
        DW      0370H           ;30.0 kΩ → 37.0
        DW      0369H           ;30.1 kΩ → 36.9
        DW      0368H           ;30.2 kΩ → 36.8
        DW      0368H           ;30.3 kΩ → 36.8
        DW      0367H           ;30.4 kΩ → 36.7
        DW      0366H           ;30.5 kΩ → 36.6
        DW      0365H           ;30.6 kΩ → 36.5
        DW      0365H           ;30.7 kΩ → 36.5
        DW      0364H           ;30.8 kΩ → 36.4
        DW      0363H           ;30.9 kΩ → 36.3
        DW      0362H           ;31.0 kΩ → 36.2
        DW      0361H           ;31.1 kΩ → 36.1
        DW      0361H           ;31.2 kΩ → 36.1
        DW      0360H           ;31.3 kΩ → 36.0
        DW      0359H           ;31.4 kΩ → 35.9
        DW      0358H           ;31.5 kΩ → 35.8
        DW      0358H           ;31.6 kΩ → 35.8
        DW      0357H           ;31.7 kΩ → 35.7
        DW      0356H           ;31.8 kΩ → 35.6
        DW      0355H           ;31.9 kΩ → 35.5
        DW      0354H           ;32.0 kΩ → 35.4
        DW      0354H           ;32.1 kΩ → 35.4
        DW      0353H           ;32.2 kΩ → 35.3
        DW      0352H           ;32.3 kΩ → 35.2
        DW      0351H           ;32.4 kΩ → 35.1
        DW      0351H           ;32.5 kΩ → 35.1
        DW      0350H           ;32.6 kΩ → 35.0
        DW      0349H           ;32.7 kΩ → 34.9
        DW      0348H           ;32.8 kΩ → 34.8
        DW      0348H           ;32.9 kΩ → 34.8
        DW      0347H           ;33.0 kΩ → 34.7
        DW      0346H           ;33.1 kΩ → 34.6
        DW      0346H           ;33.2 kΩ → 34.6
        DW      0345H           ;33.3 kΩ → 34.5
        DW      0344H           ;33.4 kΩ → 34.4
        DW      0343H           ;33.5 kΩ → 34.3
        DW      0343H           ;33.6 kΩ → 34.3
        DW      0342H           ;33.7 kΩ → 34.2
        DW      0341H           ;33.8 kΩ → 34.1
        DW      0341H           ;33.9 kΩ → 34.1
        DW      0340H           ;34.0 kΩ → 34.0
        DW      0339H           ;34.1 kΩ → 33.9
        DW      0338H           ;34.2 kΩ → 33.8
        DW      0338H           ;34.3 kΩ → 33.8
        DW      0337H           ;34.4 kΩ → 33.7
        DW      0336H           ;34.5 kΩ → 33.6
        DW      0336H           ;34.6 kΩ → 33.6
        DW      0335H           ;34.7 kΩ → 33.5
        DW      0334H           ;34.8 kΩ → 33.4
        DW      0334H           ;34.9 kΩ → 33.4
        DW      0333H           ;35.0 kΩ → 33.3
        DW      0332H           ;35.1 kΩ → 33.2
        DW      0332H           ;35.2 kΩ → 33.2
        DW      0331H           ;35.3 kΩ → 33.1
        DW      0330H           ;35.4 kΩ → 33.0
        DW      0330H           ;35.5 kΩ → 33.0
        DW      0329H           ;35.6 kΩ → 32.9
        DW      0328H           ;35.7 kΩ → 32.8
        DW      0328H           ;35.8 kΩ → 32.8
        DW      0327H           ;35.9 kΩ → 32.7
        DW      0326H           ;36.0 kΩ → 32.6
        DW      0326H           ;36.1 kΩ → 32.6
        DW      0325H           ;36.2 kΩ → 32.5
        DW      0324H           ;36.3 kΩ → 32.4
        DW      0324H           ;36.4 kΩ → 32.4
        DW      0323H           ;36.5 kΩ → 32.3
        DW      0322H           ;36.6 kΩ → 32.2
        DW      0322H           ;36.7 kΩ → 32.2
        DW      0321H           ;36.8 kΩ → 32.1
        DW      0320H           ;36.9 kΩ → 32.0
        DW      0320H           ;37.0 kΩ → 32.0
TR2HEATE:
```

In C language, the temperature conversion table is defined as in assembly language.

```
/*=================================================================

        ROM definitions

=================================================================*/
/*-----------------------------------------------------------------
        Table used to convert the resistance to a temperature
-----------------------------------------------------------------
        The temperature is referenced according to the offset based on 24.5 kΩ [100 Ω].
        BCD [0.1°C] is referenced.
-----------------------------------------------------------------*/
const unsigned short tR2Heat[] =
{
        0x0420       /* 24.5 kΩ → 42.0 */
       ,0x0419       /* 24.6 kΩ → 41.9 */
       ,0x0418       /* 24.7 kΩ → 41.8 */
       ,0x0417       /* 24.8 kΩ → 41.7 */
       ,0x0416       /* 24.9 kΩ → 41.6 */
       ,0x0415       /* 25.0 kΩ → 41.5 */
       ,0x0414       /* 25.1 kΩ → 41.4 */
       ,0x0413       /* 25.2 kΩ → 41.3 */
       ,0x0412       /* 25.3 kΩ → 41.2 */
       ,0x0411       /* 25.4 kΩ → 41.1 */
       ,0x0410       /* 25.5 kΩ → 41.0 */
       ,0x0409       /* 25.6 kΩ → 40.9 */
       ,0x0408       /* 25.7 kΩ → 40.8 */
       ,0x0407       /* 25.8 kΩ → 40.7 */
       ,0x0406       /* 25.9 kΩ → 40.6 */
       ,0x0405       /* 26.0 kΩ → 40.5 */
       ,0x0405       /* 26.1 kΩ → 40.5 */
       ,0x0404       /* 26.2 kΩ → 40.4 */
       ,0x0403       /* 26.3 kΩ → 40.3 */
       ,0x0402       /* 26.4 kΩ → 40.2 */
       ,0x0401       /* 26.5 kΩ → 40.1 */
       ,0x0400       /* 26.6 kΩ → 40.0 */
       ,0x0399       /* 26.7 kΩ → 39.9 */
       ,0x0398       /* 26.8 kΩ → 39.8 */
       ,0x0397       /* 26.9 kΩ → 39.7 */
       ,0x0396       /* 27.0 kΩ → 39.6 */
       ,0x0395       /* 27.1 kΩ → 39.5 */
       ,0x0394       /* 27.2 kΩ → 39.4 */
       ,0x0393       /* 27.3 kΩ → 39.3 */
       ,0x0392       /* 27.4 kΩ → 39.2 */
       ,0x0392       /* 27.5 kΩ → 39.2 */
       ,0x0391       /* 27.6 kΩ → 39.1 */
       ,0x0390       /* 27.7 kΩ → 39.0 */
       ,0x0389       /* 27.8 kΩ → 38.9 */
       ,0x0388       /* 27.9 kΩ → 38.8 */
       ,0x0387       /* 28.0 kΩ → 38.7 */
       ,0x0386       /* 28.1 kΩ → 38.6 */
       ,0x0385       /* 28.2 kΩ → 38.5 */
       ,0x0384       /* 28.3 kΩ → 38.4 */
       ,0x0384       /* 28.4 kΩ → 38.4 */
       ,0x0383       /* 28.5 kΩ → 38.3 */
       ,0x0382       /* 28.6 kΩ → 38.2 */
       ,0x0381       /* 28.7 kΩ → 38.1 */
       ,0x0380       /* 28.8 kΩ → 38.0 */
       ,0x0379       /* 28.9 kΩ → 37.9 */
       ,0x0378       /* 29.0 kΩ → 37.8 */
       ,0x0378       /* 29.1 kΩ → 37.8 */
       ,0x0377       /* 29.2 kΩ → 37.7 */
       ,0x0376       /* 29.3 kΩ → 37.6 */
       ,0x0375       /* 29.4 kΩ → 37.5 */
       ,0x0374       /* 29.5 kΩ → 37.4 */
       ,0x0373       /* 29.6 kΩ → 37.3 */
       ,0x0373       /* 29.7 kΩ → 37.3 */
       ,0x0372       /* 29.8 kΩ → 37.2 */
       ,0x0371       /* 29.9 kΩ → 37.1 */
       ,0x0370       /* 30.0 kΩ → 37.0 */
       ,0x0369       /* 30.1 kΩ → 36.9 */
       ,0x0368        /* 30.2 kΩ → 36.8 */
       ,0x0368        /* 30.3 kΩ → 36.8 */
       ,0x0367        /* 30.4 kΩ → 36.7 */
       ,0x0366        /* 30.5 kΩ → 36.6 */
       ,0x0365        /* 30.6 kΩ → 36.5 */
       ,0x0365        /* 30.7 kΩ → 36.5 */
       ,0x0364        /* 30.8 kΩ → 36.4 */
       ,0x0363        /* 30.9 kΩ → 36.3 */
       ,0x0362        /* 31.0 kΩ → 36.2 */
       ,0x0361        /* 31.1 kΩ → 36.1 */
       ,0x0361        /* 31.2 kΩ → 36.1 */
       ,0x0360        /* 31.3 kΩ → 36.0 */
       ,0x0359        /* 31.4 kΩ → 35.9 */
       ,0x0358        /* 31.5 kΩ → 35.8 */
       ,0x0358        /* 31.6 kΩ → 35.8 */
       ,0x0357        /* 31.7 kΩ → 35.7 */
       ,0x0356        /* 31.8 kΩ → 35.6 */
       ,0x0355        /* 31.9 kΩ → 35.5 */
       ,0x0354        /* 32.0 kΩ → 35.4 */
       ,0x0354        /* 32.1 kΩ → 35.4 */
       ,0x0353        /* 32.2 kΩ → 35.3 */
       ,0x0352        /* 32.3 kΩ → 35.2 */
       ,0x0351        /* 32.4 kΩ → 35.1 */
       ,0x0351        /* 32.5 kΩ → 35.1 */
       ,0x0350        /* 32.6 kΩ → 35.0 */
       ,0x0349        /* 32.7 kΩ → 34.9 */
       ,0x0348        /* 32.8 kΩ → 34.8 */
       ,0x0348        /* 32.9 kΩ → 34.8 */
       ,0x0347        /* 33.0 kΩ → 34.7 */
       ,0x0346        /* 33.1 kΩ → 34.6 */
       ,0x0346        /* 33.2 kΩ → 34.6 */
       ,0x0345        /* 33.3 kΩ → 34.5 */
       ,0x0344        /* 33.4 kΩ → 34.4 */
       ,0x0343        /* 33.5 kΩ → 34.3 */
       ,0x0343        /* 33.6 kΩ → 34.3 */
       ,0x0342        /* 33.7 kΩ → 34.2 */
       ,0x0341        /* 33.8 kΩ → 34.1 */
       ,0x0341        /* 33.9 kΩ → 34.1 */
       ,0x0340        /* 34.0 kΩ → 34.0 */
       ,0x0339        /* 34.1 kΩ → 33.9 */
       ,0x0338        /* 34.2 kΩ → 33.8 */
       ,0x0338        /* 34.3 kΩ → 33.8 */
       ,0x0337        /* 34.4 kΩ → 33.7 */
       ,0x0336        /* 34.5 kΩ → 33.6 */
       ,0x0336        /* 34.6 kΩ → 33.6 */
       ,0x0335        /* 34.7 kΩ → 33.5 */
       ,0x0334        /* 34.8 kΩ → 33.4 */
       ,0x0334        /* 34.9 kΩ → 33.4 */
       ,0x0333        /* 35.0 kΩ → 33.3 */
       ,0x0332        /* 35.1 kΩ → 33.2 */
       ,0x0332        /* 35.2 kΩ → 33.2 */
       ,0x0331        /* 35.3 kΩ → 33.1 */
       ,0x0330        /* 35.4 kΩ → 33.0 */
       ,0x0330        /* 35.5 kΩ → 33.0 */
       ,0x0329        /* 35.6 kΩ → 32.9 */
       ,0x0328        /* 35.7 kΩ → 32.8 */
       ,0x0328        /* 35.8 kΩ → 32.8 */
       ,0x0327        /* 35.9 kΩ → 32.7 */
       ,0x0326        /* 36.0 kΩ → 32.6 */
       ,0x0326        /* 36.1 kΩ → 32.6 */
       ,0x0325        /* 36.2 kΩ → 32.5 */
       ,0x0324        /* 36.3 kΩ → 32.4 */
       ,0x0324        /* 36.4 kΩ → 32.4 */
       ,0x0323        /* 36.5 kΩ → 32.3 */
       ,0x0322        /* 36.6 kΩ → 32.2 */
       ,0x0322        /* 36.7 kΩ → 32.2 */
       ,0x0321        /* 36.8 kΩ → 32.1 */
       ,0x0320        /* 36.9 kΩ → 32.0 */
       ,0x0320        /* 37.0 kΩ → 32.0 */
};
```

(3) Processing for setting up the peripheral functions

The following operations are performed during the processing for setting up the peripheral functions in assembly language:

<1> Interrupts are disabled.

<2> The register bank is specified.

<3> The stack pointer is specified.

<4> The memory and internal expansion RAM sizes are specified.
Specify an IMS and IXS[Note] that suit the microcontroller.

<5> The ports are set up.
P112 is set to high-level output so that it can be used as the TxD6 pin. P33 is set to low-level output so that it can be used as a port for charging the capacitor. P31 and P32 are specified as input ports during the initial settings so they can be used as discharge ports when measuring the capacitor discharge time. Other ports are set to low-level output.

<6> The clock frequencies are specified.
The CPU clock and peripheral hardware clock are specified to operate on the internal high-speed oscillation clock.

<7> 8-bit timer H2 is set up.
8-bit timer H2 is specified as a 100 ms cycle interval timer so that it can be used as the base timer for creating the timing for temperature measurement processing.

<8> 16-bit timer/event counter 00 is set up.
16-bit timer/event counter 00 is set up to measure the pulse width (in clear & start mode specified by the valid edge of the signal input to the TI000 pin) to measure the capacitor discharge pulse width. 16-bit timer capture/compare register 010 is specified to operate as a capture register and capturing the 16-bit timer capture/compare register 000 is specified to be triggered in the reverse phase of the valid edge of the TI000 pin. 16-bit timer/event counter 00 is enabled during pulse width measurement processing.

<9> The serial interface UART6 is set up as follows.

- Baud rate: 115,200 bps
- Character length of data: 8 bits
- Parity bit: Not output
- Number of stop bits: 1
- Start bit: LSB

P112 is specified as the input of the TxD6 pin and then input to this pin is enabled.

<10> Interrupts are specified to be masked.
All interrupts are masked.

<11> Interrupts are enabled.

Note These registers are provided only in the 78K0/LF3 and 78K0/LE3.

```
         ;******************************************************************
         ;
         ;
         ;       Initial settings of the peripheral functions
         ;
         ;
         ;******************************************************************
         XMAIN   CSEG    UNIT
         RESET_START:

         ;------------------------------------------------------------------
         ;      Disable interrupts
         ;------------------------------------------------------------------
<1>----         DI
         ;------------------------------------------------------------------
         ;      Specify the register bank
         ;------------------------------------------------------------------
<2>----         SEL     RB0
         ;------------------------------------------------------------------
         ;      Specify the stack pointer
         ;------------------------------------------------------------------
<3>----         MOVW    SP,     #STACKTOP
         ;------------------------------------------------------------------
         ;      Specify the ROM and RAM sizes

         ;------------------------------------------------------------------
         ;      Note that the settings differ depending on the model.
         ;      Enable the settings of the model (µPD78F0485 by default).
         ;------------------------------------------------------------------
                ;Settings when the µPD78F0471, µPD78F0481, or µPD78F0491 is used
<4>             ;MOV   IMS,   #04H               ;Specifies the ROM size.
                ;MOV   IXS,   #0CH               ;Specifies the internal expansion RAM size.

                ;Settings when the µPD78F0472, µPD78F0482, or µPD78F0492 is used
                ;MOV   IMS,   #0C6H              ;Specifies the ROM size.
                ;MOV   IXS,   #0CH               ;Specifies the internal expansion RAM size.

                ;Settings when the µPD78F0473, µPD78F0483, or µPD78F0493 is used
                ;MOV   IMS,   #0C8H              ;Specifies the ROM size.
                ;MOV   IXS,   #0CH               ;Specifies the internal expansion RAM size.

                ;Settings when the µPD78F0474, µPD78F0484, or µPD78F0494 is used
                ;MOV   IMS,   #0CCH              ;Specifies the ROM size.
                ;MOV   IXS,   #0AH               ;Specifies the internal expansion RAM size.

                ;Settings when the µPD78F0475, µPD78F0485, or µPD78F0495 is used
                MOV    IMS,   #0CFH              ;Specifies the ROM size.
                MOV    IXS,   #0AH               ;Specifies the internal expansion RAM size.

         ;------------------------------------------------------------------
         ;      Setup of port 1
         ;------------------------------------------------------------------
                MOV    P1,    #00000000B         ;Sets P1 to its initial value.
                              ;++++++++--------- P17/P16/P15/P14/P13/P12/P11/P10: Unused (0)
                MOV    PM1,   #00000000B         ;Sets P1 to input or output.
                              ;++++++++-------------------
                PM17/PM16/PM15/PM14/PM13/PM12/PM11/PM10: Unused (0)
         ;------------------------------------------------------------------
<5>      ;      Setup of port 2
         ;------------------------------------------------------------------
                MOV    P2,    #00000000B         ;Sets P2 to its initial value.
                              ;++++++++--------- P27/P26/P25/P24/P23/P22/P21/P20: Unused (0)
                MOV    PM2,   #00000000B         ;Sets P2 to input or output.
                              ;++++++++------------------
                PM27/PM26/PM25/PM24/PM23/PM22/PM21/PM20: Unused (0)
```

<5>

```
;------------------------------------------------------------------
;      Setup of port 3
;------------------------------------------------------------------
       MOV    P3,    #00000000B         ;Sets P3 to its initial value.
                     ;|||+++++--------- P33/P32/P31/P34/P30: Lo (0)
                     ;+++-------------- <Fixed to 000>
       MOV    PM3,   #11100110B         ;Sets P3 to input or output.
                     ;|||+|||+--------- PM34/PM30: Unused (0)
                     ;||| |++---------- PM32/PM31: Input (1)  Used as ports for
discharging the capacitor.
                     ;||| +------------ PM33: Output (0)  Used as a port for charging
the capacitor.
                     ;|||               (Used as TI000 when measuring the pulse
width.)
                     ;+++-------------- <Fixed to 111>
;------------------------------------------------------------------
;      Setup of port 4
;------------------------------------------------------------------
       MOV    P4,    #00000000B         ;Sets P4 to its initial value.
                     ;++++++++--------- P47/P46/P45/P44/P43/P42/P41/P40: Unused (0)
       MOV    PM4,   #00000000B         ;Sets P4 to input or output.
                     ;++++++++--------- PM47/PM46/PM45/PM44/PM43/PM42/PM41/PM40:
Unused (0)
;------------------------------------------------------------------
;      Setup of port 8
;------------------------------------------------------------------
       MOV    P8,    #00000000B         ;Sets P8 to its initial value.
                     ;||||++++--------- P83/P82/P81/P80: Unused (0)
                     ;++++------------- <Fixed to 0000>
       MOV    PM8,   #11110000B         ;Sets P8 to input or output.
                     ;||||++++--------- PM83/PM82/PM81/PM80: Unused (0)
                     ;++++------------- <Fixed to 1111>
;------------------------------------------------------------------
;      Setup of port 9
;------------------------------------------------------------------
       MOV    P9,    #00000000B         ;Sets P9 to its initial value.
                     ;||||++++--------- P93/P92/P91/P90: Unused (0)
                     ;++++------------- <Fixed to 0000>
       MOV    PM9,   #11110000B         ;Sets P9 to input or output.
                     ;||||++++--------- PM93/PM92/PM91/PM90: Unused (0)
                     ;++++------------- <Fixed to 1111>
;------------------------------------------------------------------
;      Setup of port 10
;------------------------------------------------------------------
       MOV    P10,   #00000000B         ;Sets P10 to its initial value.
                     ;||||++++--------- P103/P102/P101/P100: Unused (0)
                     ;++++------------- <Fixed to 0000>
       MOV    PM10,  #11110000B         ;Sets P10 to input or output.
                     ;||||++++--------- PM103/PM102/PM101/PM100: Unused (0)
                     ;++++------------- <Fixed to 1111>
;------------------------------------------------------------------
;      Setup of port 11
;------------------------------------------------------------------
       MOV    P11,   #00000100B         ;Sets P11 to its initial value.
                     ;||||+|++--------- P113/P111/P110: Unused (0)
                     ;|||| +----------- P112: Hi (1)
                     ;++++------------- <Fixed to 0000>
       MOV    PM11,  #11110000B         ;Sets P11 to input or output.
                     ;||||+|++--------- PM113/PM111/PM110: Unused (0)
                     ;|||| +----------- PM112: Output (0)  Used as TxD6.
                     ;++++------------- <Fixed to 1111>
;------------------------------------------------------------------
;      Setup of port 12
;------------------------------------------------------------------
       MOV    P12,   #00000000B         ;Sets P12 to its initial value.
                     ;|||||||+--------- P120: Unused (0)
                     ;|||++++---------- P124/P123/P122/P121: Read only
                     ;+++-------------- <Fixed to 000>
       MOV    PM12,  #11111110B         ;Sets P12 to input or output.
                     ;|||||||+--------- PM120: Unused (0)
                     ;+++++++---------- <Fixed to 1111111>
```

<5>

```
;-------------------------------------------------------------------
;      Setup of port 13
;-------------------------------------------------------------------
      MOV    P13,   #00000000B         ;Sets P13 to its initial value.
                    ;||||++++--------- P133/P132/P131/P130: Unused (0)
                    ;++++------------- <Fixed to 0000>
      MOV    PM13,  #11110000B         ;Sets P13 to input or output.
                    ;||||++++--------- PM133/PM132/PM131/PM130: Unused (0)
                    ;++++------------- <Fixed to 1111>
;-------------------------------------------------------------------
;      Setup of port 14
;-------------------------------------------------------------------
      MOV    P14,   #00000000B         ;Sets P14 to its initial value.
                    ;||||++++--------- P143/P142/P141/P140: Unused (0)
                    ;++++------------- <Fixed to 0000>
      MOV    PM14,  #11110000B         ;Sets P14 to input or output.
                    ;||||++++--------- PM143/PM142/PM141/PM140: Unused (0)
                    ;++++------------- <Fixed to 1111>
;-------------------------------------------------------------------
;      Setup of port 15
;-------------------------------------------------------------------
      MOV    P15,   #00000000B          ;Sets P15 to its initial value.
                    ;||||++++-----------P153/P152/P151/P150: Unused (0)
                    ;++++---------------<Fixed to 0000>
      MOV    PM15,  #11110000B          ;Sets P15 to input or output.
                    ;||||++++-----------PM153/PM152/PM151/PM150: Unused (0)
                    ;++++---------------<Fixed to 1111>
```

<6>

```
;------------------------------------------------------------------
;      Specify the clock frequency
;------------------------------------------------------------------
;      The clocks are specified to operate on the 8 MHz (TYP.) internal high-speed
oscillation clock.
;------------------------------------------------------------------
       MOV   OSCCTL,#00000000B ;Clock operating mode
                    ;||||++++--- <Fixed to 0000>
                    ;|||+------- OSCSELS: Input port mode
                    ;||+-------- <Fixed to 0>
                    ;++--------- EXCLK/OSCSEL:
                    ;          Operating mode of the high-speed system clock pin:
Input port mode
                    ;          P121/X1,P122/X2/EXCLK: Input port

       MOV   MOC,    #10000000B ;Main OSC control
                    ;|+++++++--- <Fixed to 0000000>
                    ;+---------- Stops the X1 oscillator and disables the external
clock from the EXCLK pin.

       MOV   MCM,    #00000000B ;Selects the clock to supply.
                    ;|||||+|+--- XSEL/MCM0:
                    ;||||| |   Main system clock (fXP) = Internal high-speed
oscillation clock (fRH)
                    ;||||| |   Peripheral hardware clock (fPRS) = Internal high-speed
oscillation clock (fRH)
                    ;||||| +---- MCS: Read only
                    ;+++++------ <Fixed to 00000>

       MOV   PCC,    #00000000B ;Selects the CPU clock (fCPU).
                    ;|||+|+++--- CSS/PCC2/PCC1/PCC0:
                    ;||| |     CPU clock (fCPU) = fXP
                    ;||| +------ <Fixed to 0>
                    ;||+-------- CLS: Main system clock
                    ;++--------- <Fixed to 00>

       MOV   RCM,    #00000001B ;Selects the CPU clock (fCPU).
                    ;|||||||+--- LSRSTOP: Stops the internal low-speed oscillator.
                    ;||||||+---- RSTOP: Oscillates the internal high-speed
oscillator.
                    ;|+++++----- <Fixed to 00000>
                    ;+---------- RSTS: Read only
```

<7>

```
;------------------------------------------------------------------
;       8-bit timer H2
;------------------------------------------------------------------
;       8-bit timer H2 is specified as a 100 ms interval timer and is used to measure
;       the temperature and as the interval for UART transmission (every second).
;------------------------------------------------------------------
       MOV   TMHMD2,#01100000B  ;Timer clock selection register
                    ;|||||||+------- TOEN2: Disables timer output.
                    ;||||||+--------- TOLEV2: Timer output level  Unused
                    ;||||++-------- TMMD21/TMMD20: Timer operation = Interval
                    ;|+++--------- CKS22/CKS21/CKS20:
                    ;          Count clock fPRS/2^12 (1953.125 Hz if fPRS is 8 MHz)
                    ;+------------- TMHE2: Disables timer operation.  (Enables timer
operation after the timer is set up.)

       MOV   CMP02,  #(195-1)   ;100 ms interval:(fPRS/2^12)*0.1[sec] = 195.3125

       SET1  TMHE2             ;Starts timer operation.
       CLR1  TMHIF2            ;Clears interrupt requests.

       MOV   R1SECCNT, #C1SEC  ;Initializes the 1-second counter of the TMH0 base
timer.
```

<8>

```
;------------------------------------------------------------------
;      16-bit timer/event counter 00
;------------------------------------------------------------------
;      The capacitor discharge time (pulse width) is measured to measure the
temperature sensor resistance.
;------------------------------------------------------------------
       MOV    TMC00, #00000000B  ;16-bit timer mode control register 00
                     ;|||||||+---- OVF00: Clears the TM00 overflow flag.
                     ;||||||+----- TMC001: Timer output (TO00) is inverted when
                     ;             TM00 and CR000 or TM00 and CR010 match.
                     ;||||++------ TMC003/TMC002: Disables 16-bit timer/event counter
00.
                     ;++++-------- <Fixed to 0>
       MOV    CRC00, #00000111B  ;Capture/compare control register 00
                     ;|||||||+---- CRC000: Uses CR000 as a capture register.
                     ;||||||+----- CRC001: Triggers the capturing of CR000 in the
reverse phase of the valid edge of the TI000 pin.
                     ;|||||+------ CRC002: Uses CR010 as a capture register.
                     ;+++++------- <Fixed to 0>
       MOV    TOC00, #0000000B   ;16-bit timer output control register 00
                     ;|||||||+---- TOE00: Disables TO00 output.
                     ;||||||+----- TOC001: Disables the inversion of TO00 output when
CR000 and TM00 match.
                     ;||||++------ LVS00/LVR00: The status of the TO00 pin output does
not change.
                     ;|||+-------- TOC004: Disables the inversion of TO00 output when
CR010 and TM00 match.
                     ;||+--------- OSPE00: One-shot pulse output operates as
successive pulse output.
                     ;|+---------- OSPT00: One-shot pulse output is not triggered by
software.
                     ;+----------- <Fixed to 0>
       MOV    PRM00, #00000000B  ;Prescaler mode register 00
                     ;|||||+++---- PRM002/PRM001/PRM000: Setting prohibited because
fPRS = fRH.
                     ;||||+------- <Fixed to 0>
                     ;||++-------- ES001/ES000: Valid edge of the TI000 pin: Falling
edge
                     ;++---------- ES101/ES100: Valid edge of the TI010 pin: Falling
edge
```

<9>

```
;------------------------------------------------------------------
;      UART6 setup
;------------------------------------------------------------------
;      UART6 is used to transmit the measurement result by using the temperature
sensor.
;------------------------------------------------------------------

       MOV    CKSR6, #00000000B    ;Selects the UART6 base clock.
                     ;||||++++------- TPS63-60: Base clock (fXCLK6) = fPRS
                     ;++++----------- <Fixed to 0>

       ;Specify the value to divide the baud rate clock.
       MOV    BRGC6, #35           ;Baud rate = 8*10^6[Hz]/(2 * 115200[bps]) = 34.72
                                   ;*Fractions are rounded up to minimize errors.
                                   ;Baud rate: 115200 bps ← 114285 bps (ERR: -0.79%)

       MOV    ASIM6, #01000101B    ;Selects the UART6 operating mode.
                     ;|||||||+------- ISRM6: Generates an INTSR6 interrupt when a
reception error occurs.
                     ;||||||+-------- SL6: Number of stop bits = 1
                     ;|||||+--------- CL6: Data length = 8
                     ;|||++---------- PS61-60: No parity
                     ;||+------------ RXE6: Disables reception.
                     ;|+------------- TXE6: Enables transmission.
                     ;+-------------- POWER6: Disables the internal operation clock.
```

```
            MOV    ASICL6, #00010110B   ;Selects the start bit and inverts the TxD6 output.
                          ;|||||||+------- TXDLV6: Normal TxD6 output
                          ;||||||+-------- DIR6: Start bit: LSB
                          ;|||+++--------- SBL62-60: Unused
                          ;||+------------ SBTT6: Unused
                          ;|+------------- SBRT6: Read only
                          ;+-------------- SBRF6: Unused

<9>         MOV    ISC,   #00001000B    ;Controls switching the input.
                          ;|||||||+------- ISC0: Unused
                          ;||||||+-------- ISC1: Selects the signal input from the
     P33/TI000 pin as the source of input to TI000.
                          ;|||||+--------- ISC2: Unused
                          ;||||+---------- ISC3: Enables input to RxD6/P113.
                          ;||++----------- ISC5-4: TxD6 = P112,RxD6 = P113
                          ;++------------- <Fixed to 0>

            SET1   POWER6           ;Enables the internal operation clock.


     ;------------------------------------------------------------------
     ;      Specify interrupt masking
<10> ;------------------------------------------------------------------
            MOVW     MK0,#0FFFFH
            MOVW     MK1,#0FFFFH           ;Masks all interrupts.

     ;------------------------------------------------------------------
     ;      Enable interrupts
     ;------------------------------------------------------------------
<11>        EI
```

During the initialization processing in C language, operations similar to those in assembly language are performed.

In C language, the initial settings can be performed earlier by creating the hdwinit function.

The hdwinit function is created by the user as required to set up the peripheral functions (sfr).

```
/******************************************************************

        Initialization processing after a reset release

******************************************************************/
void hdwinit(void)
{
       DI();                        /* Disables interrupts.      */

/*-----------------------------------------------------------------
       Specify the ROM and RAM sizes
------------------------------------------------------------------/
       Note that the settings differ depending on the model.
       Enable the settings of the model (μPD78F0485 by default).
-----------------------------------------------------------------*/
       /* Settings when the μPD78F0471, μPD78F0481, or μPD78F0491 is used */
       /*IMS = 0x04;              /* Specifies the ROM size. */
       /*IXS = 0x0C;              /* Specifies the internal expansion RAM size. */

       /* Settings when the μPD78F0472, μPD78F0482, or μPD78F0492 is used */
       /*IMS = 0xC6;              /* Specifies the ROM size. */
       /*IXS = 0x0C;              /* Specifies the internal expansion RAM size. */

       /* Settings when the μPD78F0473, μPD78F0483, or μPD78F0493 is used */
       /*IMS = 0xC8;              /* Specifies the ROM size. */
       /*IXS = 0x0C;              /* Specifies the internal expansion RAM size. */

       /* Settings when the μPD78F0474, μPD78F0484, or μPD78F0494 is used */
       /*IMS = 0xCC;              /* Specifies the ROM size. */
       /*IXS = 0x0A;              /* Specifies the internal expansion RAM size. */

       /* Settings when the μPD78F0475, μPD78F0485, or μPD78F0495 is used */
       IMS =   0xCF;              /* Specifies the ROM size. */
       IXS =   0x0A;              /* Specifies the internal expansion RAM size. */

/*-----------------------------------------------------------------
       Port setup (Unused ports are set to low-level output.)
-----------------------------------------------------------------*/
       /* Port 1 */
       P1 =   0b00000000;         /* Sets P1 to its initial value. */
              /*++++++++------------ P17/P16/P15/P14/P13/P12/P11/P10: Unused (0) */
       PM1 =  0b00000000;         /* Sets P1 to input or output. */
              /*++++++++------------ PM17/PM16/PM15/PM14/PM13/PM12/PM11/PM10: Unused
(0) */
       /* Port 2 */
       P2 =   0b00000000;         /* Sets P2 to its initial value. */
              /*++++++++------------ P27/P26/P25/P24/P23/P22/P21/P20: Unused (0) */
       PM2 =  0b00000000;         /* Sets P2 to input or output. */
              /*++++++++------------ PM27/PM26/PM25/PM24/PM23/PM22/PM21/PM20: Unused
(0) */
       /* Port 3 */
       P3 =   0b00000000;         /* Sets P3 to its initial value. */
              /*|||+++++------------ P33/P32/P31/P34/P30: Lo (0) */
              /*+++----------------- <Fixed to 000> */
       PM3 =  0b11100110;         /* Sets P3 to input or output. */
              /*|||+|||+------------ PM34/PM30: Unused (0) */
              /*||| |++------------- PM32/PM31: Input (1)  Used as ports for
connecting the temperature sensor. */
              /*||| +--------------- PM33: Output (0)  Used as a port for charging
the capacitor. */
              /*|||                                (Used as TI000 when measuring the
pulse width.) */
              /*+++----------------- <Fixed to 111> */
       /* Port 4 */
       P4 =   0b00000000;        /* Sets P4 to its initial value. */
              /*++++++++------------P47/P46/P45/P44/P43/P42/P41/P40: Unused (0) */
       PM4 =  0b00000000;        /* Sets P4 to input or output. */
              /*++++++++------------PM47/PM46/PM45/PM44/PM43/PM42/PM41/PM40: Unused
(0) */
       /* Port 8 */
       P8 =   0b00000000;        /* Sets P8 to its initial value. */
              /*||||++++------------P83/P82/P81/P80: Unused (0) */
              /*++++----------------<Fixed to 0000> */
       PM8 =  0b11110000;        /* Sets P8 to input or output.*/
              /*||||++++------------PM83/PM82/PM81/PM80: Unused (0) */
              /*++++----------------<Fixed to 1111> */
       /* Port 9 */
       P9 =   0b00000000;        /* Sets P9 to its initial value. */
              /*||||++++------------P93/P92/P91/P90: Unused (0) */
              /*++++----------------<Fixed to 0000> */
       PM9 =  0b11110000;        /* Sets P9 to input or output. */
              /*||||++++------------PM93/PM92/PM91/PM90: Unused (0) */
              /*++++----------------<Fixed to 1111> */
       /* Port 10 */
       P10 =  0b00000000;        /* Sets P10 to its initial value. */
              /*||||++++------------P103/P102/P101/P100: Unused (0) */
              /*++++----------------<Fixed to 0000> */
       PM10 = 0b11110000;        /* Sets P10 to input or output. */
              /*||||++++------------PM103/PM102/PM101/PM100: Unused (0) */
              /*++++----------------<Fixed to 1111> */
       /* Port 11 */
       P11 =  0b00000100;        /* Sets P11 to its initial value. */
              /*||||+|++------------P113/P111/P110: Unused (0) */
              /*|||| +--------------P112: Hi (1)*/
              /*++++---------------- <Fixed to 0000> */
       PM11 = 0b11110000;        /* Sets P11 to input or output. */
              /*||||+|++------------PM113/PM111/PM110: Unused (0) */
              /*|||| +--------------PM112: Output (0)  Used as TxD6.*/
              /*++++----------------<Fixed to 1111> */
       /* Port 12 */
       P12 =  0b00000000;        /* Sets P12 to its initial value. */
              /*|||||||+------------P120: Unused (0) */
              /*|||++++-------------P123/P122/P121: Read only */
              /*+++-----------------<Fixed to 000> */
       PM12 = 0b11111110;        /* Sets P12 to input or output. */
              /*|||||||+------------PM120: Unused (0) */
              /*+++++++-------------<Fixed to 1111111> */
       /* Port 13 */
       P13 =  0b00000000;        /* Sets P13 to its initial value. */
              /*||||++++------------P133/P132/P131/P130: Unused (0) */
              /*++++----------------<Fixed to 0000> */
       PM13 = 0b11110000;        /* Sets P13 to input or output. */
              /*||||++++------------PM133/PM132/PM131/PM130: Unused (0) */
              /*++++----------------<Fixed to 1111> */
       /* Port 14 */
       P14 =  0b00000000;        /* Sets P14 to its initial value. */
              /*||||++++------------P143/P142/P141/P140: Unused (0) */
              /*++++----------------<Fixed to 0000> */
       PM14 = 0b11110000;        /* Sets P14 to input or output. */
              /*||||++++------------PM143/PM142/PM141/PM140: Unused (0) */
              /*++++----------------<Fixed to 1111> */
       /* Port 15 */
       P15 =  0b00000000;        /* Sets P15 to its initial value. */
              /*||||++++------------P153/P152/P151/P150: Unused (0) */
              /*++++----------------<Fixed to 0000> */
       PM15 = 0b11110000;        /* Sets P15 to input or output. */
              /*||||++++------------PM153/PM152/PM151/PM150: Unused (0) */
              /*++++----------------<Fixed to 1111> */
```

```
/*-----------------------------------------------------------------
       Specify the clock frequency
-------------------------------------------------------------------
       The clocks are specified to operate on the 8 MHz (TYP.) internal high-speed
oscillation clock.
-----------------------------------------------------------------*/
       OSCCTL = 0b00000000;        /* Clock operating mode */
                /*||||++++----------- <Fixed to 0000> */
                /*|||+--------------- OSCSELS: Input port mode */
                /*||+---------------- <Fixed to 0> */
                /*++----------------- EXCLK/OSCSEL: */
                /*                      Operating mode of the high-speed system
clock pin: Input port mode */
                /*                      P121/X1,P122/X2/EXCLK: Input port */

       MOC = 0x80;                 /* Stops the X1 oscillator and disables the
external clock from the EXCLK pin. */

       MCM =    0b00000000;        /* Selects the clock to supply. */
                /*|||||+|+----------- XSEL/MCM0: */
                /*||||| |          Main system clock (fXP) = Internal high-speed
oscillation clock (fRH) */
                /*||||| |          Peripheral hardware clock (fPRS) = Internal high-
speed oscillation clock (fRH) */
                /*||||| +------------ MCS: Read only */
                /*+++++-------------- <Fixed to 00000> */

       PCC =    0b00000000;        /* Selects the CPU clock (fCPU). */
                /*|||+|+++----------- CSS/PCC2/PCC1/PCC0: */
                /*||| |                     CPU clock (fCPU) = fXP */
                /*||| +-------------- <Fixed to 0> */
                /*||+---------------- CLS: Main system clock */
                /*++----------------- <Fixed to 00> */

       RCM =    0b00000001;        /* Selects the CPU clock (fCPU). */
                /*|||||||+----------- LSRSTOP: Stops the internal low-speed
oscillator. */
                /*||||||+------------ RSTOP: Oscillates the internal high-speed
oscillator. */
                /*|+++++------------- <Fixed to 00000> */
                /*+------------------ RSTS: Read only */

/*-----------------------------------------------------------------
       8-bit timer H2
------------------------------------------------------------------/
       8-bit timer H2 is specified as a 100 ms interval timer and is used to measure
       the temperature and as the interval for UART transmission (every second).
-----------------------------------------------------------------*/
       TMHMD2 = 0b01100000;        /* Timer clock selection register */
                /* |||||||+---------- TOEN2: Disables timer output. */
                /* ||||||+----------- TOLEV2: Timer output level  Unused */
                /* ||||++------------ TMMD21/TMMD20: Timer operation = Interval */
                /* |+++-------------- CKS22/CKS21/CKS20: Count clock fPRS/2^12 */
                /* |                         (1953.125 Hz if fPRS is 8 MHz) */
                /* +----------------- TMHE2: Disables timer operation.  (Enables
timer operation after the timer is set up.) */
       CMP02 = 195-1;              /* 100 ms interval: (fPRS/2^12)*0.1[sec] =
195.3125 */

       TMHE2 = 1;                  /* Starts timer operation. */
       TMHIF2 = 0;                 /* Clears interrupt requests. */
       uc1secCnt = TMH2_1SEC;      /* Initializes the 1-second counter of the TMH0
base timer. */
```

```
/*-----------------------------------------------------------------
       16-bit timer/event counter 00
-----------------------------------------------------------------/
       The capacitor discharge time (pulse width) is measured to measure the
temperature sensor resistance.
-----------------------------------------------------------------*/
       TMC00 = 0b00000000;         /* 16-bit timer mode control register 00 */
               /*|||||||+--------- -- OVF00: Clears the TM00 overflow flag. */
               /*||||||+------------- TMC001: Timer output (TO00) is inverted when */
               /*||||||               TM00 and CR000 or TM00 and CR010 match.*/
               /*||||++-------------- TMC003/TMC002: Disables 16-bit timer/event
counter 00. */
               /*++++---------------- <Fixed to 0> */
       CRC00 = 0b00000111;         /* Capture/compare control register 00 */
               /*|||||||+------------ CRC000: Uses CR000 as a capture register. */
               /*||||||+------------- CRC001: Triggers the capturing of CR000 in the
reverse phase of the valid edge of the TI000 pin. */
               /*|||||+-------------- CRC002: Uses CR010 as a capture register. */
               /*+++++--------------- <Fixed to 0> */
       TOC00 = 0b0000000;          /* 16-bit timer output control register 00 */
               /*|||||||+------------ TOE00: Disables TO00 output. */
               /*||||||+------------- TOC001: Disables the inversion of TO00 output
when CR000 and TM00 match. */
               /*||||++-------------- LVS00/LVR00: The status of the TO00 pin output
does not change. */
               /*|||+---------------- TOC004: Disables the inversion of TO00 output
when CR010 and TM00 match. */
               /*||+----------------- OSPE00: One-shot pulse output operates as
successive pulse output. */
               /*|+------------------ OSPT00: One-shot pulse output is not triggered
by software. */
               /*+------------------- <Fixed to 0> */
       PRM00 = 0b00000000;         /* Prescaler mode register 00 */
               /*|||||+++------------ PRM002/PRM001/PRM000: Setting prohibited
because fPRS = fRH. */
               /*||||+--------------- <Fixed to 0> */
               /*||++---------------- ES001/ES000: Valid edge of the TI000 pin:
Falling edge */
               /*++------------------ ES101/ES100: Valid edge of the TI010 pin:
Falling edge */

/*-----------------------------------------------------------------
        UART6 setup
-----------------------------------------------------------------/
        UART6 is used to transmit the measurement result by using the temperature
sensor.
-----------------------------------------------------------------*/

       CKSR6 = 0b00000000;         /* Selects the UART6 base clock. */
               /*||||++++------- TPS63-60: Base clock (fXCLK6) = fPRS */
               /*++++----------- <Fixed to 0> */

       /* Specify the value to divide the baud rate clock. */
       BRGC6 = 35;       /* Baud rate = 8*10^6[Hz]/(2 * 115200[bps]) = 34.72 */
                         /* *Fractions are rounded up to minimize errors. */
                         /* Baud rate: 115200 bps ← 114285 bps (ERR: -0.79%) */

       ASIM6 = 0b01000101;         /* Selects the UART6 operating mode. */
               /*|||||||+------------ ISRM6: Generates an INTSR6 interrupt when a
reception error occurs. */
               /*||||||+------------- SL6: Number of stop bits = 1 */
               /*|||||+-------------- CL6: Data length = 8 */
               /*|||++--------------- PS61-60: No parity */
               /*||+----------------- RXE6: Disables reception. */
               /*|+------------------ TXE6: Enables transmission. */
               /*+------------------- POWER6: Disables the internal operation clock.
*/
```

```
       ASICL6 =0b00010110;         /* Selects the start bit and inverts the TxD6
output. */
               /*|||||||+------------ TXDLV6: Normal TxD6 output */
               /*||||||+------------- DIR6: Start bit: LSB */
               /*|||+++-------------- SBL62-60: Unused */
               /*||+----------------- SBTT6: Unused */
               /*|+------------------ SBRT6: Read only */
               /*+------------------- SBRF6: Unused */

       ISC =    0b00001000;        /* Controls switching the input. */
               /*|||||||+------------ ISC0: Unused */
               /*||||||+------------- ISC1: Selects the signal input from the
P33/TI000 pin as the source of input to TI000. */
               /*|||||+-------------- ISC2: Unused */
               /*||||+--------------- ISC3: Enables input to RxD6/P113. */
               /*||++---------------- ISC5-4: TxD6 = P112,RxD6 = P113 */
               /*++------------------ <Fixed to 0> */

       POWER6 =         1;        /* Enables the internal operation clock. */

/*-----------------------------------------------------------------
        Specify interrupt masking
-----------------------------------------------------------------*/
       MK0 = 0x0FFFF;
       MK1 = 0x0FFFF;             /* Masks all interrupts. */

       EI();                      /* Enables interrupts */
}
```

5.2 Main Processing

The following operations are performed during the main processing in assembly language:

<1> The timing for measuring the temperature for about one second is counted by using 8-bit timer H2 as the base timer. After about one second has elapsed, <2> to <5> are performed.

<2> The capacitor discharge time is measured by using a fixed resistor for calibration. The use of the fixed resistor for calibration is specified in the pulse width measurement mode and pulse width measurement processing is called.

<3> The capacitor discharge time is measured by using a thermistor. The use of the thermistor is specified in the pulse width measurement mode and pulse width measurement processing is called.

<4> Temperature acquisition processing is called.

<5> UART transmission processing is called.

<6> The main processing branches to <1>.

```
        ;*******************************************************************
        ;
        ;
        ;       Main processing
        ;
        ;
        ;*******************************************************************
        MAIN_LOOP:
               ;*****************************************************;
               ;                                                     ;
               ;  Processing to transmit the measured temperature    ;
               ;                                                     ;
               ;*****************************************************;
               ;-----------------------------------;
               ;     Timing creation processing    ;
      ┌-----   ;-----------------------------------;
      |  LMAIN100:
      |        BF      TMHIF2,$LMAIN500      ;Have 100 ms elapsed? → NO
  <1> |        CLR1    TMHIF2                ;Clears interrupt requests.
      |        DEC     R1SECCNT              ;Updates the 1-second counter.
      |        BNZ     $LMAIN500             ;Has 1 second elapsed? → NO
      |        MOV     R1SECCNT,#C1SEC       ;Initializes the 1-second counter.
      └-----
               ;------------------------------------;
               ; Temperature measurement processing ;
               ;------------------------------------;
      ┌-----
      |        MOV      B,#0            ;Specifies an argument (pulse width measurement mode).
  <2> |        CALL     !SGETPULSE      ;Measures the discharge pulse width of the fixed
      |  resistor for calibration.
      |        MOVW     RCALBCNT,AX     ;Acquires the measured pulse width.
      └-----
      ┌-----
      |        MOV      B,#1            ;Specifies an argument (pulse width measurement mode).
  <3> |        CALL     !SGETPULSE      ;Measures the discharge pulse width of the thermistor.
      |        MOVW     RTHERMCNT,AX    ;Acquires the measured pulse width.
      └-----

         LMAIN400:
  <4>----      CALL     !SGETHEAT       ; Calculates the resistance from the measured pulse
         width and acquires the temperature.

               ;-----------------------------------;
               ;     Creation and transmission of UART6 data ;
               ;-----------------------------------;
  <5>----      CALL    !SUART6TX

         LMAIN500:
               ;***********************************;
               ;                                   ;
               ;      Different types of main processing     ;
               ;                                   ;
               ;***********************************;

               ; Any other main processing is performed here.

  <6>----      BR      MAIN_LOOP
```

During the main processing in C language, operations similar to those in assembly language are performed.

```
/******************************************************************

        Main loop

******************************************************************/
void main(void)
{
        while(1)
        {
             /***************************************************/
             /*                                                 */
             /* Processing to transmit the measured temperature */
             /*                                                 */
             /***************************************************/
             /*----------------------------------*/
             /*    Timing creation processing    */
             /*----------------------------------*/
                if(TMHIF2)
                {/* 100 ms has elapsed.  */
                        TMHIF2 = 0;      /* Clears interrupt requests. */
                        uc1secCnt--;     /* Updates the 1-second counter. */
                }

             /*------------------------------------*/
             /* Temperature measurement processing */
             /*------------------------------------*/
                if(uc1secCnt == 0)
                {/* 1 second has elapsed. */
                        uc1secCnt = TMH2_1SEC;      /* Clears the 1-second counter. */

                        /* Measures the discharge pulse width of the fixed resistor
for calibration. */
                        ushCalibrationCnt = fn_GetPulseTime(0);

                        /* Measures the discharge pulse width of the thermistor. */
                        ushThermistorCnt = fn_GetPulseTime(1);

                        /* Calculates the resistance from the measured pulse width
and acquires the temperature. */
                        ushHeatData = fn_GetHeatData();

                        /* Creation and transmission of UART6 data */
                        fn_UART6_Tx();
                }
             /**************************************/
             /*                                    */
             /* Different types of main processing */
             /*                                    */
             /**************************************/

                /* Any other main processing is performed here. */
        }
}
```

5.3 Pulse Width Measurement Processing

The following operations are performed during the pulse width measurement processing in assembly language:

<1> The pulse width measurement mode is saved in register A, because register B is used in <3>.

<2> The counter that counts overflows is initialized.

<3> The capacitor is charged. P33 is set to high-level output and the system waits 2 ms.

<4> P33 is specified as an input port so that it can be used as the TI000 pin.

<5> Requests to generate an interrupt when the valid edge of the TI000 pin is detected are cleared.

<6> Discharging the capacitor is started by setting the discharge port specified in pulse width measurement mode (P31 or P32)Note to low-level output.

(a) At the same time, the enabling operation of 16-bit timer/event counter 00 is set to clear & start mode by the valid edge of the signal input to the TI000 pin and measuring the discharge time starts.

<7> The system waits until the capacitor discharges and TI000 goes to low level. Next, the interrupt generated when the valid edge of the TI000 pin is detected (INTTM010) is generated and the value of 16-bit timer counter 00 is captured by 16-bit timer capture/compare register 010.

(a) If 16-bit timer counter 00 overflows while measuring the discharge time, the counter that counts overflows is updated. The fixed resistor for calibration and capacitor, which are used to measure the discharge time by using a fixed resistor for calibration, are selected such that 16-bit timer counter 00 will not overflow. If at least two overflows occur when the discharge time is measured by using a thermistor, the result of calculating the thermistor resistance will be outside the measurement range. Therefore, the occurrence of at least two overflows is considered as a measurement error and measuring the capacitor discharge time is suspended.

<8> The capacitor discharge time is acquired from 16-bit timer capture/compare register 010.

<9> The used discharge port (P31 or P32)Note is specified as an input port.

(a) The fixed resistor for calibration and capacitor, which are used to measure the discharge time by using a fixed resistor for calibration, are selected such that 16-bit timer counter 00 will not overflow. Therefore, if an overflow occurs when measuring the discharge pulse width for calibration, the discharge time is set to 0 due to a measurement error.

<10> Operation of 16-bit timer/event counter 00, which was enabled, is disabled.

<11> P33 is specified as an input port.

Note This port is P31 when using a fixed resistor for calibration to discharge the capacitor and P32 when using a thermistor.

```
        ;******************************************************************
        ;
        ;       Measurement of the capacitor discharge time (measurement of the TI000 pulse
        width)
        ;
        ;------------------------------------------------------------------
        ;       [  IN  ] B: Pulse width measurement mode (0: The discharge pulse width of the
        fixed resistor for calibration is measured.
        ;                                                 1: The discharge pulse width of the
        thermistor is measured.)
        ;       [  OUT ] AX: Measured discharge pulse width
        ;                ROVFCNT: Number of times TM00 overflows

        ;
        ;       The capacitor discharge time is measured by determining the pulse width by
        using TI000.
        ;       Whether to measure the discharge pulse width of a fixed resistor for
        calibration
        ;       or a thermistor is specified by using an argument.
        ;       The measured discharge pulse width is returned.
        ;       If TM00 overflows while measuring the pulse width,
        ;       the number of overflows is set to the appropriate counter.
        ;******************************************************************
        SGETPULSE:
<1>----         MOV     A,B                     ;Acquires the pulse width measurement mode.
<2>----         MOV     ROVFCNT,#0              ;Clears the counter that counts overflows.

                ;============    Charge the capacitor    ============
                SET1    P3.3
                CLR1    PM3.3                   ;Starts charging the capacitor.
                ;Waits 2 ms for the capacitor to charge.
<3>             MOV     B,#93       ;[4clk]                     ↑
        JGETP100:                   ;                           |
                MOV     C,#27       ;[4clk] ↑                   |4 + (166 + 6)*93 = 16000clk
        JGETP101:                   ;       |4+6*27 = 166clk    |16000 * 0.125[μs] = 2000[μs]
                DBNZ    C,$JGETP101 ;[6clk] ↓                   |
                DBNZ    B,$JGETP100 ;[6clk]                     ↓

<4>----         SET1    PM3.3                 ;Uses P33 as TI000.
<5>----         CLR1    TMIF010               ;Clears interrupt requests.

                ;===========   Start discharging the capacitor   ===========
                MOV     B,A                   ;Saves the pulse width measurement mode.
                CMP     A,#0                  ;Has the discharge pulse width of the fixed
        resistor for calibration been measured?
                BNZ     $JGETP200             ;→ NO: The discharge pulse width of the
        thermistor is measured.

                CLR1    P3.1                  ;Prepares to discharge.  Starts discharging when
        P31 is set to low-level output.
<6> (a) -       MOV     TMC00,#08H            ;Starts measuring the pulse width by using 16-bit
        timer/event counter 00.
                CLR1    PM3.1                 ;Starts discharging.
                BR      JGETP300
        JGETP200:
                CLR1    P3.2                  ;Prepares to discharge.   Starts discharging when
        P32 is set to low-level output.
    (a) -       MOV     TMC00,#08H            ;Starts measuring the pulse width by using 16-bit
        timer/event counter 00.
                CLR1    PM3.2                 ;Starts discharging.

        JGETP300:
                ;==========  Wait for the capacitor to discharge  ==========
                BT      TMIF010,$JGETP500     ;Has the capacitor discharged? → YES

                BF      OVF00,$JGETP400       ;Is TM00 overflow detected? → NO
                CLR1    OVF00                 ;Clears the TM00 overflow flag.
    (a)         INC     ROVFCNT               ;Updates the number of overflows.
                CMP     ROVFCNT,#2            ;Have at least 2 overflows occurred?
<7>             BZ      $JGETP500             ;→ YES: A temperature measurement error occurs
        and pulse width measurement is suspended.
        JGETP400:
                BR      JGETP300              ;The wait for the capacitor to discharge
        continues.

        JGETP500:
                CLR1    TMIF010               ;Clears interrupt requests.
                ;===========   Finish discharging the capacitor   ===========
<8>             MOVW   AX,CR010             ;Acquires the measured pulse width.
                DEC    B
                BZ     $JGETP700            ;Has the discharge pulse width of the fixed
        resistor for calibration been measured?
<9>                                         ;→ NO: The discharge pulse width of the
        thermistor is measured.
                SET1   PM3.1                ;Sets the port used to discharge the capacitor
        when using the fixed resistor for calibration back to input.
                CMP    ROVFCNT,#0           ;Has an overflow occurred when measuring the pulse
(a)     width?
                BZ     $JGETP800            ;→ NO: Returns the measured pulse width.
                MOVW   AX,#0                ;Returns the value as an error.
                BR     JGETP800
        JGETP700:
                SET1   PM3.2                ;Sets the port used to discharge the capacitor
        when using a thermistor back to input.

        JGETP800:
<10>            MOV    TMC00,#0             ;Stops 16-bit timer/event counter 00.
<11>            CLR1   P3.3
                CLR1   PM3.3                ;Sets TI000 back to low-level output.

                RET
```

During the processing in C language, operations similar to those in assembly language are performed.

```
/*********************************************************************************

      Measurement of the capacitor discharge time (measurement of the TI000 pulse
width)

----------------------------------------------------------------------------------
        [  IN  ] mode (0: The discharge pulse width of the fixed resistor for
calibration is measured.
                      1: The discharge pulse width of the thermistor is measured.)
        [  OUT ] Measured discharge pulse width

        The capacitor discharge time is measured by determining the pulse width by
using TI000.
        Whether to measure the discharge pulse width of a fixed resistor for
calibration
        or a thermistor is specified by using an argument.
        The measured discharge pulse width is returned.
        If TM00 overflows while measuring the pulse width,
        the number of overflows is set to the appropriate counter.
*********************************************************************************/
static short fn_GetPulseTime(unsigned char mode)
{
        unsigned short ushRet;     /* Used to save the return value. */
        unsigned short temp;       /* Work area */

        ucOVFcnt = 0;        /* Clears the counter that counts overflows. */

        /* Charge the capacitor */
        P3.3 = 1;
        PM3.3 = 0;           /* Starts charging the capacitor. */
        for(temp = 224; temp > 0; temp--)
        {
                NOP();       /* Waits about 2 ms for the capacitor to charge. */
        }
        PM3.3 = 1;           /* Uses P33 as TI000. */
        TMIF010 = 0;         /* Clears interrupt requests. */

        /* Start discharging the capacitor */
        if(mode == 0)
        {/* Measurement of the discharge pulse width of the fixed resistor for
calibration */
                P3.1 = 0;          /* Prepares to discharge. *//* Starts discharging
when P31 is set to low-level output. */
                TMC00 = 0x08;      /* Starts measuring the pulse width. */
                PM3.1 = 0;         /* Starts discharging. */
        }
        else
        {/* Measurement of the discharge pulse width of the thermistor */
                P3.2 = 0;          /* Prepares to discharge. *//* Starts discharging
when P32 is set to low-level output. */
                TMC00 = 0x08;      /* Starts measuring the pulse width. */
                PM3.2 = 0;         /* Starts discharging. */
        }

        /* Wait for the capacitor to discharge */
        while(!TMIF010)
        {
                if(OVF00)
                {/* If an overflow of TM00 has been detected */
                        OVF00 = 0;      /* Clears the TM00 overflow flag. */
                        ucOVFcnt++;     /* Updates the number of overflows. */
                        if(ucOVFcnt >= 2)    /* If at least 2 overflows have occurred
*/
                                break;  /* A temperature measurement error occurs and
pulse width measurement is suspended. */
                }
        }
        TMIF010 = 0;         /* Clears interrupt requests. */
        ushRet = CR010;      /* Acquires the measured pulse width. */
        /* Finish discharging the capacitor */
        if(mode == 0){    /* Sets the port used to discharge the capacitor back to
input. */
                PM3.1 = 1;    /* If a fixed resistor for calibration is used */
                if(ucOVFcnt > 0)
                        ushRet = 0;
        }
        else
        {
                PM3.2 = 1;    /* If a thermistor is used */
        }

        TMC00 = 0x00;     /* Stops 16-bit timer/event counter 00. */
        P3.3 = 0;
        PM3.3 = 0;        /* Sets TI000 back to low-level output. */

        return ushRet;    /* Returns the pulse width.*/
}
```

5.4 Temperature Acquisition Processing

The following operations are performed during the temperature acquisition processing in assembly language:

<1> Whether errors occurred while measuring the capacitor discharge time is checked. If the discharge time measured by using a fixed resistor for calibration is invalid, or at least two overflows occurred while measuring the discharge pulse width by using a thermistor, a measurement error occurs. If no measurement error occurred, <2> to <4> are performed. If a measurement error occurred, <5> is performed.

<2> The thermistor resistance is calculated. Equation 1 is expanded to equation 2 and operations are performed in the order of (a), (b), and then (c).

$$R_{TH} = \frac{R_C \times (CNT_{TH} + \text{number of overflows} \times 10000H)}{CNT_C} \quad \cdots\cdots \text{[Equation 1]}$$

$$R_{TH} = \underbrace{\underbrace{\{ \underbrace{(R_C \times CNT_{TH})}_{(a)} + (R_C \times \text{number of overflows} \times 10000H) \}}_{(b)} \div CNT_C}_{(c)} \quad \cdots\cdots \text{[Equation 2]}$$

R_{TH}: Thermistor resistance [100 Ω]

R_C: Resistance of the fixed resistor for calibration [100 Ω]

CNT_{TH}: Capacitor discharge time if a thermistor is used

CNT_C: Capacitor discharge time if a fixed resistor for calibration is used

<3> Whether the thermistor resistance calculated in <2> is within the resistance measurement range (24.5 to 37.0 kΩ) with respect to the temperature measurement range (42.0 to 32.0°C) is determined. If the thermistor resistance is outside the range, <5> is performed.

<4> The temperature corresponding to the thermistor resistance is acquired from the temperature conversion table[Note]. The temperature (BCD) is acquired by calculating the offset (in 100 Ω units) of the thermistor resistance from the minimum resistance (24.5 kΩ) in the measurement range and then converting that offset to the offset from the start address in the temperature conversion table.

<5> An error occurs, and the result of measuring the temperature is set to FFFFH.

<6> Function of the multiplication performed in <2>

<7> Function of the division performed in <2>

Note For details about the temperature conversion table, see **2.2 Converting Resistance to Temperature**.

```
;*******************************************************************
;
;       Temperature acquisition processing
;
;-------------------------------------------------------------------
;       [  IN  ]       RCALBCNT: Discharge pulse width of the fixed resistor for
calibration
;                      RTHERMCNT: Discharge pulse width of the thermistor
;                      ROVFCNT: Number of times TM00 overflows (when measuring the
discharge pulse width of the thermistor)
;       [  OUT ]       RHEAT: Temperature (BCD)
;
;       The resistance is calculated from the measured pulse width and
;       the temperature is acquired from the temperature conversion table.
;
;    ◎The resistance is calculated from the pulse width by using the following
equation:
;
;                 Rc × (CNTth + number of overflows x 10000H)
;         Rth =----------------------------------------------
;                                         CNTc
;
;               Rth: Thermistor resistance [100 Ω]
;               Rc: Resistance of the fixed resistor for calibration = 330 [100 Ω]
;               CNTth: Discharge pulse width of the thermistor
;               CNTc: Discharge pulse width of the fixed resistor for calibration
;
;
;      ◎The value relative to the Rth measurement range is calculated by using the
equation below,
;         and the temperature is acquired from the temperature conversion table by
using that value as the offset.
;
;               Rrel = Rth - Rmin
;
;               Rrel:  Value relative to the Rth measurement range [100 Ω]
;               Rmin:  Minimum resistance in the measurement range = 245 [100 Ω]
;*******************************************************************
SGETHEAT:
        CMP   RCALBCNT,#0        ;Did an error occur while measuring the discharge
pulse width of the fixed resistor for calibration?
        BZ    $JGETH800          ; → YES: The resistance cannot be calculated.  The
<1> resistance is not calculated.

        CMP   ROVFCNT,#2         ;Did at least two overflows occur while measuring the
pulse width?
        BZ    $JGETH800          ; → YES: The resistance is already outside the
measurement range.  The resistance is not calculated.

        ;---------------------------------------------------;
        ;   Calculate the resistance from the pulse width    ;
        ;---------------------------------------------------;
        MOVW  RTEMP32,#0         ;Saves 0 to the lower 16 bits of the variable used
for calculation.
        MOVW  AX,RTHERMCNT
        MOVW  (RTEMP32+2),AX     ;Saves CNTth to the higher 16 bits of the variable
(a) used for calculation.
        MOVW  AX,#330
        MOVW  RTEMP16A,AX        ;Saves Rc (330) [100 Ω] to the variable used for
<2> calculation.
        CALL  !SMULT16           ;Calculates (Rc x CNTth).

        ;Adds (Rc x number of overflows x 10000H) to the result of (Rc x CNTth).
        CMP   ROVFCNT,#0         ;Has an overflow occurred?
        BZ    $JGETH300          ; → NO: Calculating the resistance continues without
adding 10000H.
(b)     MOV   A,ROVFCNT
        MOV   B,A                ;Sets the number of overflows to the counter.
        MOVW  AX,(RTEMP32+2)     ;Adds Rc to the higher 16 bits of the result of (Rc x
CNTth).
```

```
         JGETH200:
                 ADDW  AX,#330           ;Adds Rc.
     (b)         BC    $JGETH800         ;Has the result of addition overflowed? → YES: The
<2>      temperature is outside the measurement range.
                 DBNZ  B,$JGETH200       ;Has Rc been added for the number of overflows? → NO
                 MOVW  (RTEMP32+2),AX

         JGETH300:
                 MOVW  AX,RCALBCNT
     (c)         MOVW  RTEMP16A,AX       ;Specifies the discharge pulse width of the fixed
         resistor for calibration as the divisor.
                 CALL  !SDIV32           ;Calculates (Rc x (CNTth + number of overflows x
         10000H)/CNTc.


                 ;-----------------------------------------------------;
                 ;   Determine whether the thermistor resistance is within the measurement
         range (24.5 kΩ to 37.0 kΩ)   ;
                 ;-----------------------------------------------------;
                 MOVW    AX,(RTEMP32+2)  ;Acquires the higher 16 bits of the calculated
         resistance.
                 CMPW    AX,#0000H       ;Compares the higher 16 bits with 0 (based on 370 =
         172H).
                 BNZ     $JGETH800       ;If the higher 16 bits are at least 1, the
         temperature is identified as an error, because the resistance is outside the
<3>      measurement range.
         JGETH400:
                 MOVW    AX,RTEMP32      ;Acquires the lower 16 bits of the calculated
         resistance.
                 CMPW    AX,#371         ;Is the calculated resistance 37.0 kΩ or less?
                 BNC     $JGETH800       ; → NO: The temperature is identified as an error.
                 CMPW    AX,#245         ;Is the calculated resistance at least 24.5 kΩ?
                 BC      $JGETH800       ; → NO: The temperature is identified as an error.

                 ;-----------------------------------;
                 ;   Convert the resistance to a temperature   ;
                 ;-----------------------------------;
         JGETH500:       ;Calculates the value relative to the Rth measurement range.
                 MOVW    AX,RTEMP32
                 SUBW    AX,#245         ;Calculates Rrel = Rth − Rmin.
                 MOV     A,X             ;Acquires the lower 8 bits.  (If Rrel is within the
         measurement range, Rrel falls within the 8 bits.)
                 ADD     A,A             ;Doubles Rrel and
                 MOV     B,A             ;acquires the offset in the temperature conversion
         table.
<4>              MOVW    HL,#TR2HEAT     ;Sets the address in the temperature conversion table
         to HL.
                 MOV     A,[HL+B]        ;Acquires the temperature (lower 8 bits).
                 MOV     X,A
                 INC     B
                 MOV     A,[HL+B]        ;Acquires the temperature (higher 8 bits).
                 MOVW    RHEAT,AX        ;Saves the temperature to a variable.
                 BR      JGETH900

                 ;-----------------------------------------------------;
                 ;   Temperature setting if an error occurred while measuring the
         temperature  ;
                 ;-----------------------------------------------------;
         JGETH800:
<5>              MOVW    RHEAT,#0FFFFH   ;Identifies the temperature as an error.

         JGETH900:
                 RET
```

<6>

```
;*****************************************************************
;
;       Function used for multiplication (16 bits * 16 bits)
;
;-----------------------------------------------------------------
;       [  IN  ] RTEMP16A: Multiplier
;                RTEMP32: Saves the multiplier to the higher 16 bits and 0 to the
lower 16 bits.
;       [  OUT ] RTEMP32: Operation result
;*****************************************************************
SMULT16:
        MOV    B,#16                ;Sets up the bit counter.
JMLT120:
        CLR1   CY
        MOV    A,RTEMP32
        ROLC   A,1
        MOV    RTEMP32,A
        MOV    A,(RTEMP32+1)
        ROLC   A,1
        MOV    (RTEMP32+1),A
        MOV    A,(RTEMP32+2)
        ROLC   A,1
        MOV    (RTEMP32+2),A
        MOV    A,(RTEMP32+3)
        ROLC   A,1
        MOV    (RTEMP32+3),A        ;Left-shifts the operation result (including the
multiplicand) 1 bit.
        BNC    $JMLT220             ;MSB = 1? → NO

        MOV    A,RTEMP16A
        ADD    A,RTEMP32
        MOV    RTEMP32,A
        MOV    A,(RTEMP16A+1)
        ADDC   A,(RTEMP32+1)
        MOV    (RTEMP32+1),A
        MOV    A,#0
        ADDC   A,RTEMP16A
        MOV    RTEMP16A,A           ;Adds the multiplicand.
JMLT220:
        DBNZ   B,$JMLT120           ;Have 16 bits been processed? → NO

        RET
```

<7>

```
;*******************************************************************
;
;       Function used for division (32 bits/16 bits)
;
;-------------------------------------------------------------------
;       [  IN  ] RTEMP16A: Divisor
;                RTEMP32: Dividend
;       [  OUT ] RTEMP32: Operation result
;                RTEMP16B: Remainder
;*******************************************************************
SDIV32:
        MOVW   RTEMP16B,#0         ;Initializes the variable used for calculation.

        MOV    B,#32               ;Sets up the bit counter.
JDIV120:
        CLR1   CY
        MOV    A,RTEMP16B
        ROLC   A,1
        MOV    RTEMP16B,A
        MOV    A,(RTEMP16B+1)
        ROLC   A,1
        MOV    (RTEMP16B+1),A
        MOV    A,RTEMP32
        ROLC   A,1
        MOV    RTEMP32,A
        MOV    A,(RTEMP32+1)
        ROLC   A,1
        MOV    (RTEMP32+1),A
        MOV    A,(RTEMP32+2)
        ROLC   A,1
        MOV    (RTEMP32+2),A
        MOV    A,(RTEMP32+3)
        ROLC   A,1
        MOV    (RTEMP32+3),A       ;Left-shifts the dividend 1 bit.
        MOV    A,#0
        ADDC   A,RTEMP16B
        MOV    RTEMP16B,A          ;MSB -> LSB

        SUB    A,RTEMP16A
        MOV    RTEMP16B,A
        MOV    A,(RTEMP16B+1)
        SUBC   A,(RTEMP16A+1)
        MOV    (RTEMP16B+1),A      ;RTEMP16B - RTEMP16A
        BT     RTEMP32.0,$JDIV220  ;Is borrowing possible? → YES
        BC     $JDIV180            ;RTEMP16B < RTEMP16A ? → YES

        SET1   RTEMP32.0           ;Specifies the quotient.
        BR     JDIV220

JDIV180:
        MOV    A,RTEMP16B
        ADD    A,RTEMP16A
        MOV    RTEMP16B,A
        MOV    A,(RTEMP16B+1)
        ADDC   A,(RTEMP16A+1)
        MOV    (RTEMP16B+1),A
JDIV220:
        DBNZ   B,$JDIV120          ;Have 32 bits been processed? → NO

        RET
```

During the processing in C language, operations similar to those in assembly language are performed.

```
/*******************************************************************

        Temperature acquisition processing

-------------------------------------------------------------------
        [  IN  ] None
        [  OUT ] Temperature (BCD)

        The resistance is calculated from the measured pulse width and
        the temperature is acquired from the temperature conversion table.

      ◎The resistance is calculated from the pulse width by using the following
equation:
                (assuming that the resistance and pulse width are proportional)
                Rc : CNTc = Rth : CNTth

                          Rc × (CNTth + number of overflows x 0x10000)
                → Rth = ------------------------------------------------
                                             CNTc

                Rth:   Thermistor resistance [100 Ω]
                Rc:    Resistance of the fixed resistor for calibration = 330 [100 Ω]
                CNTth: Discharge pulse width of the thermistor
                CNTc:  Discharge pulse width of the fixed resistor for calibration


       ◎The value relative to the Rth measurement range is calculated by using the
equation below,
          and the temperature is acquired from the temperature conversion table by
using that value as the offset.

                Rrel = Rth - Rmin

                Rrel: Value relative to the Rth measurement range [100 Ω]
                Rmin: Minimum resistance in the measurement range = 245 [100 Ω]
******************************************************************/
static short fn_GetHeatData(void)
{
        unsigned short ushRet;         /* Used to save the return value. */
        unsigned long int ulTemp1;     /* RAM used for calculation */
        unsigned char ucTemp2;         /* RAM used for calculation */

        if((ushCalibrationCnt != 0) && (ucOVFcnt < 2))
        {/* If the discharge pulse width of the fixed resistor for calibration can be
measured */
         /* and no more than two overflows occur while measuring the discharge pulse
width of the thermistor resistance, */
         /* the resistance is calculated from the pulse width. */
                /* The measured thermistor pulse width is expanded to 32 bits by
adding the overflow portion. */
                ulTemp1 = (unsigned long)(ucOVFcnt * 0x10000) + ushThermistorCnt;
                /* The thermistor resistance is calculated. */
                ushRet = (unsigned short)((ulTemp1 * 330) / ushCalibrationCnt);

                /* Whether the thermistor resistance is within the measurement range
(24.5 kΩ to 37.0 kΩ) is determined. */
                if((ushRet <= 370)&&(ushRet >= 245))
                {/* If the resistance is within the measurement range, the
temperature is acquired from the resistance. */
                        ucTemp2 = (unsigned char)(ushRet - 245);
                        ushRet = tR2Heat[ucTemp2];
                }
                else
                {/* If the resistance is outside the measurement range, the
temperature is identified as an error. */
                        ushRet = 0xffff;
                }
        }
        else
        {/* If at least two overflows occurred while measuring the thermistor
discharge pulse width,  */
        /* the resistance is already outside the measurement range. */
                ushRet = 0xffff;                /* The temperature is identified as
an error. */
        }

        return ushRet;          /* Returns the temperature. */
}
```

5.5 UART Transmission Processing

The following operations are performed during the UART transmission processing in assembly language:

<1> The result of measuring the temperature is converted to ASCII code.[Note]

<2> The result of measuring the temperature converted to ASCII code is transmitted via the serial interface UART6.

Note For details about the UART communication settings and the transmitted data, see **4.4 UART Data Transmission Format**.

```
;******************************************************************
;
;       Creation and transmission of UART6 data
;
;------------------------------------------------------------------
;       [  IN  ] RHEAT: Temperature (BCD)
;       [  OUT ] None
;
;       The measured temperature is converted to ASCII code, set to the transmit
buffer,
;       and then transmitted.
;
;       <Example of transmitted data>
;              ◎ If 38.5°C was measured
;                 0   1   2   3   4   5
;
;                 3   8   .   5   ¥r  ¥n
;
;              ◎ If an error occurred while measuring the temperature
;                 0   1   2   3   4   5
;
;                 *   *   .   *   ¥r  ¥n
;
;
;******************************************************************
SUARTT6TX:
        ;------------------------------------------
        ;   Processing to create UART6 transmit data in the transmit buffer
        ;------------------------------------------
        MOVW    AX,RHEAT                ;Has the temperature been measured?
        CMPW    AX,#0FFFFH              ; → NO
        BZ      $JU6TX100

        ;The temperature is set to the transmit buffer.
        AND     A,#0FH                  ;Acquires the 10s digit.
        ADD     A,#'0'                  ;Converts the value to ASCII code.
        MOV     RTXBUF,A                ;[0]Saves the 10s digit of the temperature.

        MOV     A,X                     ;Acquires the 1s and tenth digits.
<1>     ROR     A,1                     ;Shifts the higher 4 bits to the lower 4 bits.
        ROR     A,1
        ROR     A,1
        ROR     A,1
        AND     A,#0FH                  ;Acquires the 1s digit of the lower 4 bits.
        ADD     A,#'0'                  ;Converts the value to ASCII code.
        MOV     (RTXBUF+1),A            ;[1]Saves the 1s digit of the temperature.

        MOV     (RTXBUF+2),#'.'         ;[2]Saves the decimal point.
        MOV     (RTXBUF+2),#'.'      ;[2]Saves the decimal point.

        MOV     A,X                 ;Acquires the 1s and tenth digits.
        AND     A,#0FH              ;Acquires the tenth digit.
        ADD     A,#'0'              ;Converts the value to ASCII code.
        MOV     (RTXBUF+3),A  ;[3]Saves the tenth digit of the temperature.

<1>     BR      JU6TX200

JU6TX100:
        ;Sets **.* to the transmit buffer.
        MOV     RTXBUF,#'*'          ;[0]Saves the asterisk.
        MOV     (RTXBUF+1),#'*'      ;[1]Saves the asterisk.
        MOV     (RTXBUF+2),#'.'      ;[2]Saves the decimal point.
        MOV     (RTXBUF+3),#'*'      ;[3]Saves the asterisk.

JU6TX200:
        MOV     (RTXBUF+4),#0DH      ;[4]Saves the carriage return.
        MOV     (RTXBUF+5),#0AH      ;[5]Saves the line feed.

        ;------------------------------------------
        ;          UART6 data transmission
        ;------------------------------------------
JU6TX500:
        ;=============    Start transmission    =============
        MOV     B,#6                ;Sets up the transmission counter.
        MOVW    HL,#RTXBUF

JU6TX600:
<2>     CLR1    STIF6               ;Clears interrupt requests.
        MOV     A,[HL]              ;Acquires transmit data from the transmit buffer.
        MOV     TXB6,A              ;Transmits the data.
JU6TX700:
        BF      STIF6,$JU6TX700     ;Has 1 byte been transmitted via UART6? → NO
        CLR1    STIF6               ;Clears interrupt requests.
        INCW    HL                  ;Updates the location of the transmit data in the
transmit buffer.
        DBNZ    B,$JU6TX600         ;Is there data not transmitted? → YES: The next
data unit is transmitted.
JU6TX800:               ;Transmission ends.
        RET
```

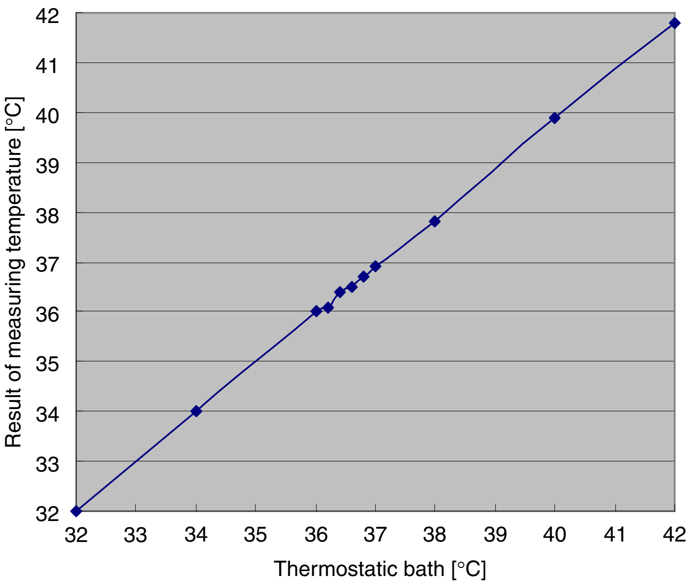
During the processing in C language, operations similar to those in assembly language are performed.

```
/*******************************************************************

        Creation and transmission of UART6 data

-------------------------------------------------------------------
        [  IN  ] None
        [  OUT ] None

        The measured temperature is converted to ASCII code, set to the transmit
buffer,
        and then transmitted.

        <Example of transmitted data>
               ◎If 38.5°C was measured
                   0   1   2   3   4   5
                 +---+---+---+---+---+---+
                 | 3 | 8 | . | 5 |¥r |¥n |
                 +---+---+---+---+---+---+

               ◎If an error occurred while measuring the temperature
                   0   1   2   3   4   5
                 +---+---+---+---+---+---+
                 | * | * | . | * |¥r |¥n |
                 +---+---+---+---+---+---+

*******************************************************************/
static void fn_UART6_Tx(void)
{
        /***************************************/
        /*                                     */
        /*   Creation of UART6 transmit data   */
        /*                                     */
        /***************************************/
        if(ushHeatData != 0xFFFF)
        {/* If the temperature has been measured, the temperature is set to the
transmit buffer. */
                 /* [0]10s digit of the temperature (which is converted to ASCII
code) */
                ucTxBuffer[0] = (unsigned char)(((ushHeatData >> 8) & 0x000f) + '0');
                 /* [1]1s digit of the temperature (which is converted to ASCII code)
*/
                ucTxBuffer[1] = (unsigned char)(((ushHeatData >> 4) & 0x000f) + '0');
                 /* [2]Decimal point */
                ucTxBuffer[2] = '.';
                 /* [3]Tenth digit of the temperature (which is converted to ASCII
code) */
                ucTxBuffer[3] = (unsigned char)((ushHeatData & 0x000f) + '0');
        }
        else
        {/* If a measurement error occurs, **.* is set to the transmit buffer. */
                ucTxBuffer[0] = '*';    /* [0]Saves the asterisk. */
                ucTxBuffer[1] = '*';    /* [1]Saves the asterisk. */
                ucTxBuffer[2] = '.';    /* [2]Saves the decimal point. */
                ucTxBuffer[3] = '*';    /* [3]Saves the asterisk. */
        }
        ucTxBuffer[4] = '¥r';           /* [4]Carriage return */
        ucTxBuffer[5] = '¥n';           /* [5]Line feed */

        /***************************************/
        /*                                     */
        /*         UART6 data transmission     */
        /*                                     */
        /***************************************/
        for(ucTxBufferCounter = 0; ucTxBufferCounter < sizeof(ucTxBuffer);
ucTxBufferCounter++)
        {/* Transmission continues until all data has been transmitted. */
                STIF6 = 0;                              /* Clears interrupt requests.
*/
                TXB6 = ucTxBuffer[ucTxBufferCounter];   /* Transmits the data. */
                while(!STIF6)/* The system waits until 1 byte has been transmitted
via UART6. */
                        NOP();
        }
}
```

CHAPTER 6 EXAMPLE OF CHECKING OPERATION OF DEVICE

This chapter shows examples of measured temperatures. The temperatures of a thermostatic bath and the results of measuring the temperature when the device operates by using a thermostatic bath are summarized below.

Thermostatic Bath (°C)	Result of Measuring Temperature (°C)
32	32
34	34
36	36
36.2	36.1
36.4	36.4
36.6	36.5
36.8	36.7
37	36.9
38	37.8
40	39.9
42	41.8



CHAPTER 7 RELATED DOCUMENTS

Document Name		English
78K0/LC3 User’s Manual		PDF
78K0/LD3 User’s Manual		PDF
78K0/LE3 User’s Manual		PDF
78K0/LF3 User’s Manual		PDF
78K/0 Series Instructions User’s Manual		PDF
RA78K0 Assembler Package User’s Manual	Language	PDF
	Operation	PDF
CC78K0 C Compiler User’s Manual	Language	PDF
	Operation	PDF
PM+ Project Manager User’s Manual		PDF

APPENDIX A PROGRAM LIST

The 78K0/LF3 microcontroller source program is shown below as a program list example.

- main.asm (assembly language version)

```
;*********************************************************************
;
;       NEC Electronics     78K0/Lx3 Series
;
;*********************************************************************
;       78K0/LF3 Series        Sample program
;*********************************************************************
;   Temperature Measurement Program Using Port and Timer Functions
;*********************************************************************
; [History]
;       2008.05.--      Newly created
;*********************************************************************
;
; [Overview]
;
;This sample program measures the temperature by using an externally connected thermistor.
;The temperature is measured every second and transmitted via the serial interface UART6.
;To measure the temperature, the capacitor discharge time is separately measured by using a
;fixed resistor for calibration and a thermistor, and then the thermistor resistance is
;calculated by using the ratio between the discharge time and resistance.  The calculated
;thermistor resistance is converted to a temperature by using the temperature conversion table.
;The discharge time is measured by determining the pulse width by using 16-bit timer/event
;counter 00.
;The measurement range is from 32.0°C to 42.0°C.  If a value outside this rage is measured,
;an error is transmitted via the UART.
;The UART used in this sample program performs only transmission.
;
;
; <Primary initial settings>
;
; • Setting up the vector table
; • Specifying the register bank
; • Specifying the stack pointer
; • Specifying the ROM and RAM sizes
; • Setting up the ports
; • Specifying that the CPU clock operate on the internal high-speed oscillation clock (8 MHz
(TYP.))
; • Setting up 16-bit timer/event counter 00
; • Setting up 8-bit timer H2
; • Setting up the serial interface UART6
; • Specifying interrupt masking
```

```
;
;
; <Primary main processing>
;
; • Processing to acquire the discharge pulse width of the fixed resistor for calibration
; • Processing to acquire the discharge pulse width of the thermistor
; • Processing to acquire the temperature
; • Processing to create UART6 transmit data and transmitting the data
;
;
; <Primary processing for measuring the discharge pulse width>
;
; • Charging the capacitor
; • Starting to discharge the capacitor
; • Acquiring the TM00 count value (pulse width)
;
;
; <Primary processing to acquire the temperature>
;
; • Calculating the thermistor resistance from the pulse width
; • Identifying the thermistor resistance as an error
; • Acquiring the temperature from the thermistor resistance
;
;
; <Primary processing to create UART6 transmit data and transmitting the data>
;
; • Creating data to transmit
; • Starting communication
; • Counting the transmit data
; • Setting up the transmit data
;
;*********************************************************

;===============================================================================
;
;       Setting up the vector table
;
;===============================================================================
TVCT1   CSEG    AT      000000H
        DW      RESET_START                     ;(00)   RESET input, POC, LVI, WDT, TRAP
TVCT2   CSEG    AT      000004H
        DW      RESET_START                     ;(04)   INTLVI
        DW      RESET_START                     ;(06)   INTP0
        DW      RESET_START                     ;(08)   INTP1
        DW      RESET_START                     ;(0A)   INTP2
        DW      RESET_START                     ;(0C)   INTP3
        DW      RESET_START                     ;(0E)   INTP4
        DW      RESET_START                     ;(10)   INTP5
        DW      RESET_START                     ;(12)   INTSRE6
        DW      RESET_START                     ;(14)   INTSR6
        DW      RESET_START                     ;(16)   INTST6
        DW      RESET_START                     ;(18)   INTCSI10/INTST0
        DW      RESET_START                     ;(1A)   INTTMH1
        DW      RESET_START                     ;(1C)   INTTMH0
        DW      RESET_START                     ;(1E)   INTTM50
        DW      RESET_START                     ;(20)   INTTM000
        DW      RESET_START                     ;(22)   INTTM010
        DW      RESET_START                     ;(24)   INTAD
        DW      RESET_START                     ;(26)   INTSR0
        DW      RESET_START                     ;(28)   INTRTC
        DW      RESET_START                     ;(2A)   INTTM51
        DW      RESET_START                     ;(2C)   INTKR
        DW      RESET_START                     ;(2E)   INTRTCI
        DW      RESET_START                     ;(30)   INTDSAD
        DW      RESET_START                     ;(32)   INTTM52
        DW      RESET_START                     ;(34)   INTTMH2
        DW      RESET_START                     ;(36)   INTMCG
        DW      RESET_START                     ;(38)   INTRIN
        DW      RESET_START                     ;(3A)   INTRERR/INTGP/INTREND/INTDFULL
        DW      RESET_START                     ;(3C)   INTACSI
        DW      RESET_START                     ;(3E)   BRK

;===============================================================================
;
;       Securing the stack area
;
;===============================================================================
DSTK            DSEG    AT      0FB00H          ;First RAM address
STACKEND:
                DS      20H                     ;Secures a 32 MB stack area.
STACKTOP:                                       ;The first address of the stack area is FB20H.

;===============================================================================
;
;       RAM definitions
;
;===============================================================================
DTHERMO         DSEG    SADDR
R1SECCNT:       DS      1               ;Counts the time by using 100 ms (TMH2) as the base
timer.
 C1SEC          EQU     (1000/100)      ;Used to count 1 second.

ROVFCNT:        DS      1               ;Counter that counts the number of times TM00 overflows

RTXBUF:         DS      6               ;Transmit data buffer
```

```
DTHERMOP        DSEG    SADDRP  ;RAM related to measuring the temperature
RCALBCNT:       DS      2               ;Used to acquire the value for measuring the TI000
pulse width for calibration.
RTHERMCNT:      DS      2               ;Used to acquire the value for measuring the TI000
pulse width for the thermistor.
RHEAT:          DS      2               ;Saves the calculated temperature.  * FFFFH is output
for a measurement error.

RTEMP16A:       DS      2               ;Variable used to calculate the resistance
RTEMP16B:       DS      2               ;Variable used to calculate the resistance
RTEMP32:        DS      4               ;Variable used to calculate the resistance

;===============================================================================
;
;       ROM definitions
;
;===============================================================================
CREGACC CSEG    UNITP
;-------------------------------------------------------------------------------
;       Table used to convert the resistance to a temperature
;--------------------------------------------------------------------------------------
;  The temperature is referenced according to the offset based on 24.5 kΩ [100 Ω].
;  BCD [0.1°C] is referenced.
;--------------------------------------------------------------------------------------
TR2HEAT:
        DW      0420H           ;24.5 kΩ → 42.0
        DW      0419H           ;24.6 kΩ → 41.9
        DW      0418H           ;24.7 kΩ → 41.8
        DW      0417H           ;24.8 kΩ → 41.7
        DW      0416H           ;24.9 kΩ → 41.6
        DW      0415H           ;25.0 kΩ → 41.5
        DW      0414H           ;25.1 kΩ → 41.4
        DW      0413H           ;25.2 kΩ → 41.3
        DW      0412H           ;25.3 kΩ → 41.2
        DW      0411H           ;25.4 kΩ → 41.1
        DW      0410H           ;25.5 kΩ → 41.0
        DW      0409H           ;25.6 kΩ → 40.9
        DW      0408H           ;25.7 kΩ → 40.8
        DW      0407H           ;25.8 kΩ → 40.7
        DW      0406H           ;25.9 kΩ → 40.6
        DW      0405H           ;26.0 kΩ → 40.5
        DW      0405H           ;26.1 kΩ → 40.5
        DW      0404H           ;26.2 kΩ → 40.4
        DW      0403H           ;26.3 kΩ → 40.3
        DW      0402H           ;26.4 kΩ → 40.2
        DW      0401H           ;26.5 kΩ → 40.1
        DW      0400H           ;26.6 kΩ → 40.0
        DW      0399H           ;26.7 kΩ → 39.9
        DW      0398H           ;26.8 kΩ → 39.8
        DW      0397H           ;26.9 kΩ → 39.7
        DW      0396H           ;27.0 kΩ → 39.6
        DW      0395H           ;27.1 kΩ → 39.5
        DW      0394H           ;27.2 kΩ → 39.4
        DW      0393H           ;27.3 kΩ → 39.3
        DW      0392H           ;27.4 kΩ → 39.2
        DW      0392H           ;27.5 kΩ → 39.2
        DW      0391H           ;27.6 kΩ → 39.1
        DW      0390H           ;27.7 kΩ → 39.0
        DW      0389H           ;27.8 kΩ → 38.9
        DW      0388H           ;27.9 kΩ → 38.8
        DW      0387H           ;28.0 kΩ → 38.7
        DW      0386H           ;28.1 kΩ → 38.6
        DW      0385H           ;28.2 kΩ → 38.5
        DW      0384H           ;28.3 kΩ → 38.4
        DW      0384H           ;28.4 kΩ → 38.4
        DW      0383H           ;28.5 kΩ → 38.3
        DW      0382H           ;28.6 kΩ → 38.2
        DW      0381H           ;28.7 kΩ → 38.1
        DW      0380H           ;28.8 kΩ → 38.0
        DW      0379H           ;28.9 kΩ → 37.9
        DW      0378H           ;29.0 kΩ → 37.8
        DW      0378H           ;29.1 kΩ → 37.8
        DW      0377H           ;29.2 kΩ → 37.7
        DW      0376H           ;29.3 kΩ → 37.6
        DW      0375H           ;29.4 kΩ → 37.5
        DW      0374H           ;29.5 kΩ → 37.4
        DW      0373H           ;29.6 kΩ → 37.3
        DW      0373H           ;29.7 kΩ → 37.3
        DW      0372H           ;29.8 kΩ → 37.2
        DW      0371H           ;29.9 kΩ → 37.1
        DW      0370H           ;30.0 kΩ → 37.0
        DW      0369H           ;30.1 kΩ → 36.9
        DW      0368H           ;30.2 kΩ → 36.8
        DW      0368H           ;30.3 kΩ → 36.8
        DW      0367H           ;30.4 kΩ → 36.7
        DW      0366H           ;30.5 kΩ → 36.6
        DW      0365H           ;30.6 kΩ → 36.5
        DW      0365H           ;30.7 kΩ → 36.5
        DW      0364H           ;30.8 kΩ → 36.4
        DW      0363H           ;30.9 kΩ → 36.3
        DW      0362H           ;31.0 kΩ → 36.2
        DW      0361H           ;31.1 kΩ → 36.1
        DW      0361H           ;31.2 kΩ → 36.1
        DW      0360H           ;31.3 kΩ → 36.0
        DW      0359H           ;31.4 kΩ → 35.9
        DW      0358H           ;31.5 kΩ → 35.8
        DW      0358H           ;31.6 kΩ → 35.8
        DW      0357H           ;31.7 kΩ → 35.7
        DW      0356H           ;31.8 kΩ → 35.6
        DW      0355H           ;31.9 kΩ → 35.5
        DW      0354H           ;32.0 kΩ → 35.4
        DW      0354H           ;32.1 kΩ → 35.4
        DW      0353H           ;32.2 kΩ → 35.3
        DW      0352H           ;32.3 kΩ → 35.2
        DW      0351H           ;32.4 kΩ → 35.1
        DW      0351H           ;32.5 kΩ → 35.1
        DW      0350H           ;32.6 kΩ → 35.0
        DW      0349H           ;32.7 kΩ → 34.9
        DW      0348H           ;32.8 kΩ → 34.8
        DW      0348H           ;32.9 kΩ → 34.8
        DW      0347H           ;33.0 kΩ → 34.7
        DW      0346H           ;33.1 kΩ → 34.6
        DW      0346H           ;33.2 kΩ → 34.6
        DW      0345H           ;33.3 kΩ → 34.5
        DW      0344H           ;33.4 kΩ → 34.4
        DW      0343H           ;33.5 kΩ → 34.3
        DW      0343H           ;33.6 kΩ → 34.3
        DW      0342H           ;33.7 kΩ → 34.2
        DW      0341H           ;33.8 kΩ → 34.1
        DW      0341H           ;33.9 kΩ → 34.1
        DW      0340H           ;34.0 kΩ → 34.0
        DW      0339H           ;34.1 kΩ → 33.9
        DW      0338H           ;34.2 kΩ → 33.8
        DW      0338H           ;34.3 kΩ → 33.8
        DW      0337H           ;34.4 kΩ → 33.7
        DW      0336H           ;34.5 kΩ → 33.6
        DW      0336H           ;34.6 kΩ → 33.6
        DW      0335H           ;34.7 kΩ → 33.5
        DW      0334H           ;34.8 kΩ → 33.4
        DW      0334H           ;34.9 kΩ → 33.4
        DW      0333H           ;35.0 kΩ → 33.3
        DW      0332H           ;35.1 kΩ → 33.2
        DW      0332H           ;35.2 kΩ → 33.2
        DW      0331H           ;35.3 kΩ → 33.1
        DW      0330H           ;35.4 kΩ → 33.0
        DW      0330H           ;35.5 kΩ → 33.0
        DW      0329H           ;35.6 kΩ → 32.9
        DW      0328H           ;35.7 kΩ → 32.8
        DW      0328H           ;35.8 kΩ → 32.8
        DW      0327H           ;35.9 kΩ → 32.7
        DW      0326H           ;36.0 kΩ → 32.6
        DW      0326H           ;36.1 kΩ → 32.6
        DW      0325H           ;36.2 kΩ → 32.5
        DW      0324H           ;36.3 kΩ → 32.4
        DW      0324H           ;36.4 kΩ → 32.4
        DW      0323H           ;36.5 kΩ → 32.3
        DW      0322H           ;36.6 kΩ → 32.2
        DW      0322H           ;36.7 kΩ → 32.2
        DW      0321H           ;36.8 kΩ → 32.1
        DW      0320H           ;36.9 kΩ → 32.0
        DW      0320H           ;37.0 kΩ → 32.0
TR2HEATE:

;*******************************************************************************
;
;
;       Initial settings of the peripheral functions
;
;
;*******************************************************************************
XMAIN   CSEG    UNIT
RESET_START:

;-------------------------------------------------------------------------------
;       Disable interrupts
;-------------------------------------------------------------------------------
        DI
;-------------------------------------------------------------------------------
;       Specify the register bank
;-------------------------------------------------------------------------------
        SEL     RB0
;-------------------------------------------------------------------------------
;       Specify the stack pointer
;-------------------------------------------------------------------------------
        MOVW    SP,     #STACKTOP
;-------------------------------------------------------------------------------
;       Specify the ROM and RAM sizes

;-------------------------------------------------------------------------------
;       Note that the settings differ depending on the model.
;       Enable the settings of the model (μPD78F0485 by default).
;-------------------------------------------------------------------------------
        Settings when the μPD78F0471, μPD78F0481, or μPD78F0491 is used
        ;MOV    IMS,    #04H                    ;Specifies the ROM size.
        ;MOV    IXS,    #0CH                    ;Specifies the internal expansion RAM size.

        Settings when the μPD78F0472, μPD78F0482, or μPD78F0492 is used
        ;MOV    IMS,    #0C6H                   ;Specifies the ROM size.
        ;MOV    IXS,    #0CH                    ;Specifies the internal expansion RAM size.
        ;Settings when the μPD78F0473, μPD78F0483, or μPD78F0493 is used
        ;MOV    IMS,    #0C8H                   ;Specifies the ROM size.
        ;MOV    IXS,    #0CH                    ;Specifies the internal expansion RAM size.

        ;Settings when the μPD78F0474, μPD78F0484, or μPD78F0494 is used
        ;MOV    IMS,    #0CCH                   ;Specifies the ROM size.
        ;MOV    IXS,    #0AH                    ;Specifies the internal expansion RAM size.

        ;Settings when the μPD78F0475, μPD78F0485, or μPD78F0495 is used
        MOV     IMS,    #0CFH                   ;Specifies the ROM size.
        MOV     IXS,    #0AH                    ;Specifies the internal expansion RAM size.

;-------------------------------------------------------------------------------
;       Setup of port 1
;-------------------------------------------------------------------------------
        MOV     P1,     #00000000B              ;Sets P1 to its initial value.
                        ;++++++++--------------- P17/P16/P15/P14/P13/P12/P11/P10: Unused (0)
        MOV     PM1,    #00000000B              ;Sets P1 to input or output.
                        ;++++++++---------------PM17/PM16/PM15/PM14/PM13/PM12/PM11/PM10:  Unused
(0)
;-------------------------------------------------------------------------------
;       Setup of port 2
;-------------------------------------------------------------------------------
        MOV     P2,     #00000000B              ;Sets P2 to its initial value.
                        ;++++++++--------------- P27/P26/P25/P24/P23/P22/P21/P20: Unused (0)
        MOV     PM2,    #00000000B              ;Sets P2 to input or output.
                        ;++++++++-------------- PM27/PM26/PM25/PM24/PM23/PM22/PM21/PM20:  Unused
(0)
;-------------------------------------------------------------------------------
;       Setup of port 3
;-------------------------------------------------------------------------------
        MOV     P3,     #00000000B              ;Sets P3 to its initial value.
                        ;|||+++++--------------- P33/P32/P31/P34/P30:Lo(0)
                        ;+++-------------------- <Fixed to 000>
        MOV     PM3,    #11100110B              ;Sets P3 to input or output.
                        ;|||+|||+--------------- PM34/PM30: Unused (0)
                        ;||| |++---------------- PM32/PM31: Input (1)  Used as ports for
discharging the capacitor.
                        ;||| +------------------ PM33: Output (0)  Used as a port for charging
the capacitor.  (Used as TI000 when measuring the pulse width.)
                        ;+++-------------------- <Fixed to 111>
;-------------------------------------------------------------------------------
;       Setup of port 4
;-------------------------------------------------------------------------------
        MOV     P4,     #00000000B              ;Sets P4 to its initial value.
                        ;++++++++---------------- P47/P46/P45/P44/P43/P42/P41/P40: Unused (0)
        MOV     PM4,    #00000000B              ;Sets P4 to input or output.
                        ;++++++++---------------- PM47/PM46/PM45/PM44/PM43/PM42/PM41/PM40:
```

```
Unused (0)
;-------------------------------------------------------------------------------
;       Setup of port 8
;-------------------------------------------------------------------------------
        MOV     P8,     #00000000B              ;Sets P8 to its initial value.
                        ;||||++++--------------- P83/P82/P81/P80: Unused (0)
                        ;++++------------------- <Fixed to 0000>
        MOV     PM8,    #11110000B              ;Sets P8 to input or output.
                        ;||||++++--------------- PM83/PM82/PM81/PM80: Unused (0)
                        ;++++------------------- <Fixed to 1111>
;-------------------------------------------------------------------------------
;       Setup of port 9
;-------------------------------------------------------------------------------
        MOV     P9,     #00000000B              ;Sets P9 to its initial value.
                        ;||||++++--------------- P93/P92/P91/P90: Unused (0)
                        ;++++------------------- <Fixed to 0000>
        MOV     PM9,    #11110000B              ;Sets P9 to input or output.
                        ;||||++++--------------- PM93/PM92/PM91/PM90: Unused (0)
                        ;++++------------------- <Fixed to 1111>
;-------------------------------------------------------------------------------
;       Setup of port 10
;-------------------------------------------------------------------------------
        MOV     P10,    #00000000B              ;Sets P10 to its initial value.
                        ;||||++++--------------- P103/P102/P101/P100: Unused (0)
                        ;++++------------------- <Fixed to 0000>
        MOV     PM10,   #11110000B              ;Sets P10 to input or output.
                        ;||||++++--------------- PM103/PM102/PM101/PM100: Unused (0)
                        ;++++------------------- <Fixed to 1111>
;-------------------------------------------------------------------------------
;       Setup of port 11
;-------------------------------------------------------------------------------
        MOV     P11,    #00000100B              ;Sets P11 to its initial value.
                        ;||||+|++--------------- P113/P111/P110: Unused (0)
                        ;|||| +----------------- P112:Hi(1)
                        ;++++------------------- <Fixed to 0000>
        MOV     PM11,   #11110000B              ;Sets P11 to input or output.
                        ;||||+|++--------------- PM113/PM111/PM110: Unused (0)
                        ;|||| +----------------- PM112: Output (0)  Used as TxD6.
                        ;++++------------------- <Fixed to 1111>
;-------------------------------------------------------------------------------
;       Setup of port 12
;-------------------------------------------------------------------------------
        MOV     P12,    #00000000B              ;Sets P12 to its initial value.
                        ;|||||||+--------------- P120: Unused (0)
                        ;|||++++---------------- P124/P123/P122/P121:Read Only
                        ;+++-------------------- <Fixed to 000>
        MOV     PM12,   #11111110B              ;Sets P12 to input or output.
                        ;|||||||+--------------- PM120: Unused (0)
                        ;+++++++---------------- <Fixed to 1111111>
;-------------------------------------------------------------------------------
;       Setup of port 13
;-------------------------------------------------------------------------------
        MOV     P13,    #00000000B              ;Sets P13 to its initial value.
                        ;||||++++--------------- P133/P132/P131/P130: Unused (0)
                        ;++++------------------- <Fixed to 0000>
        MOV     PM13,   #11110000B              ;Sets P13 to input or output.
                        ;||||++++--------------- PM133/PM132/PM131/PM130: Unused (0)
                        ;++++------------------- <Fixed to 1111>
;-------------------------------------------------------------------------------
;       Setup of port 14
;-------------------------------------------------------------------------------
        MOV     P14,    #00000000B              ;Sets P14 to its initial value.
                        ;||||++++--------------- P143/P142/P141/P140: Unused (0)
                        ;++++------------------- <Fixed to 0000>
        MOV     PM14,   #11110000B              ;Sets P14 to input or output.
                        ;||||++++--------------- PM143/PM142/PM141/PM140: Unused (0)
                        ;++++------------------- <Fixed to 1111>
;-------------------------------------------------------------------------------
;       Setup of port 15
;-------------------------------------------------------------------------------
        MOV     P15,    #00000000B              ;Sets P15 to its initial value.
                        ;||||++++--------------- P153/P152/P151/P150: Unused (0)
                        ;++++------------------- <Fixed to 0000>
        MOV     PM15,   #11110000B              ;Sets P15 to input or output.
                        ;||||++++--------------- PM153/PM152/PM151/PM150: Unused (0)
                        ;++++------------------- <Fixed to 1111>

;-------------------------------------------------------------------------------
;       Specify the clock frequency
;-------------------------------------------------------------------------------
;       The clocks are specified to operate on the 8 MHz (TYP.) internal high-speed oscillation
clock.
;-------------------------------------------------------------------------------
        MOV     OSCCTL, #00000000B              ;Clock operating mode
                        ;||||++++--------------- <Fixed to 0000>
                        ;|||+------------------- OSCSELS: Input port mode
                        ;||+-------------------- <Fixed to 0>
                        ;++--------------------- EXCLK/OSCSEL:
                        ;                        Operating mode of the high-speed system clock
pin: Input port mode
                        ;                        P121/X1,P122/X2/EXCLK: Input port

        MOV     MOC,    #10000000B              ;Main OSC control
                        ;|+++++++--------------- <Fixed to 0000000>
                        ;+---------------------- Stops the X1 oscillator and disables the
external clock from the EXCLK pin.
```

```
        MOV     MCM,    #00000000B              ;Selects the clock to supply.
                        ;|||||+|+--------------- XSEL/MCM0:
                        ;||||| |                 Main system clock (fXP) = Internal high-speed
oscillation clock (fRH)
                        ;||||| |                Peripheral hardware clock (fPRS) = Internal
high-speed oscillation clock (fRH)
                        ;||||| +---------------- MCS: Read Only
                        ;+++++------------------ <Fixed to 00000>

        MOV     PCC,    #00000000B              ;Selects the CPU clock (fCPU).
                        ;|||+|+++--------------- CSS/PCC2/PCC1/PCC0:
                        ;||| |                            CPU clock (fCPU) = fXP
                        ;||| +------------------ <Fixed to 0>
                        ;||+-------------------- CLS: Main system clock
                        ;++--------------------- <Fixed to 00>

        MOV     RCM,    #00000001B              ;Selects the CPU clock (fCPU).
                        ;|||||||+--------------- LSRSTOP: Stops the internal low-speed
oscillator.
                        ;||||||+---------------- RSTOP: Oscillates the internal high-speed
oscillator.
                        ;|+++++----------------- <Fixed to 00000>
                        ;+---------------------- RSTS: Read Only

;-------------------------------------------------------------------------------
;       8-bit timer H2
;-------------------------------------------------------------------------------
;       8-bit timer H2 is specified as a 100 ms interval timer and is used to measure
;       the temperature and as the interval for UART transmission (every second).
;-------------------------------------------------------------------------------
        MOV     TMHMD2,#01100000B               ; Timer clock selection register
                        ;|||||||+--------------- TOEN2: Disables timer output.
                        ;||||||+---------------- TOLEV2: Timer output level  Unused
                        ;||||++----------------- TMMD21/TMMD20: Timer operation = Interval
                        ;|+++------------------- CKS22/CKS21/CKS20: Count clock fPRS/2^12
(1953.125 Hz if fPRS is 8 MHz)
                      ;+---------------------- TMHE2: Disables timer operation.  (Enables timer
operation after the timer is set up.)


        MOV     CMP02,  #(195-1)        ;100 ms interval: (fPRS/2^12)*0.1[sec] = 195.3125

        SET1    TMHE2                   ;Starts timer operation.
        CLR1    TMHIF2                  ;Clears interrupt requests.

        MOV     R1SECCNT,#C1SEC         ;Initializes the 1-second counter of the TMH0 base
timer.
```

```
;-------------------------------------------------------------------------------
;       16-bit timer/event counter 00
;-------------------------------------------------------------------------------
;       The capacitor discharge time (pulse width) is measured to measure the temperature
sensor resistance.
;-------------------------------------------------------------------------------
        MOV     TMC00,  #00000000B              ;16-bit timer mode control register 00
                        ;|||||||+--------------- OVF00: Clears the TM00 overflow flag.
                        ;||||||+---------------- TMC001: Timer output (TO00) is inverted when
TM00 and CR000 or TM00 and CR010 match.
                        ;||||++----------------- TMC003/TMC002: Disables 16-bit timer/event
counter 00.
                        ;++++------------------- <Fixed to 0>
        MOV     CRC00,  #00000111B              ;Capture/compare control register 00
                        ;|||||||+--------------- CRC000: Uses CR000 as a capture register.
                        ;||||||+---------------- CRC001: Triggers the capturing of CR000 in the
reverse phase of the valid edge of the TI000 pin.
                        ;|||||+----------------- CRC002: Uses CR010 as a capture register.
                        ;+++++------------------ <Fixed to 0>
        MOV     TOC00,  #0000000B               ;16-bit timer output control register 00
                        ;|||||||+--------------- TOE00: Disables TO00 output.
                        ;||||||+---------------- TOC001: Disables the inversion of TO00 output
when CR000 and TM00 match.
                        ;||||++----------------- LVS00/LVR00: The status of the TO00 pin output
does not change.
                        ;|||+------------------- TOC004: Disables the inversion of TO00 output
when CR010 and TM00 match.
                        ;||+-------------------- OSPE00: One-shot pulse output operates as
successive pulse output.
                        ;|+--------------------- OSPT00: One-shot pulse output is not triggered
by software.
                        ;+---------------------- <Fixed to 0>
        MOV     PRM00,  #00000000B              ;Prescaler mode register 00
                        ;|||||+++--------------- PRM002/PRM001/PRM000: Setting prohibited
because fPRS = fRH.
                        ;||||+------------------ <Fixed to 0>
                        ;||++------------------- ES001/ES000: Valid edge of the TI000 pin:
Falling edge
                        ;++--------------------- ES101/ES100: Valid edge of the TI010 pin:
Falling edge

;-------------------------------------------------------------------------------
;       UART6 setup
;-------------------------------------------------------------------------------
;       UART6 is used to transmit the measurement result by using the temperature sensor.
;-------------------------------------------------------------------------------
```

```
        MOV     CKSR6,  #00000000B              ; Selects the UART6 base clock.
                        ;||||++++--------------- TPS63-60: Base clock (fXCLK6) = fPRS
                        ;++++------------------- <Fixed to 0>

        ; Specify the value to divide the baud rate clock.
        MOV     BRGC6,  #35             ;Baud rate = 8*10^6[Hz]/(2 * 115200[bps]) = 34.72
                                        ;*Fractions are rounded up to minimize errors.
                                        ;Baud rate: 115200 bps ← 114285 bps (ERR: -0.79%)

        MOV     ASIM6,  #01000101B              ;Selects the UART6 operating mode.
                        ;|||||||+--------------- ISRM6: Generates an INTSR6 interrupt when a
reception error occurs.
                        ;||||||+---------------- SL6: Number of stop bits = 1
                        ;|||||+----------------- CL6: Data length = 8
                        ;|||++------------------ PS61-60: No parity
                        ;||+-------------------- RXE6: Disables reception.
                        ;|+--------------------- TXE6: Enables transmission.
                        ;+---------------------- POWER6: Disables the internal operation clock.

        MOV     ASICL6, #00010110B              ;Selects the start bit and inverts the TxD6
output.
                        ;|||||||+--------------- TXDLV6: Normal TxD6 output
                        ;||||||+---------------- DIR6: Start bit: LSB
                        ;|||+++----------------- SBL62-60: Unused
                        ;||+-------------------- SBTT6: Unused
                        ;|+--------------------- SBRT6: Read Only
                        ;+---------------------- SBRF6: Unused

        MOV     ISC,    #00001000B              ; Controls switching the input.
                        ;|||||||+--------------- ISC0: Unused
                        ;||||||+---------------- ISC1: Selects the signal input from the
P33/TI000 pin as the source of input to TI000.
                        ;|||||+----------------- ISC2: Unused
                        ;||||+------------------ ISC3: Enables input to RxD6/P113.
                        ;||++------------------- ISC5-4: TxD6=P112,RxD6=P113
                        ;++--------------------- <Fixed to 0>

        SET1    POWER6                          ;Enables the internal operation clock.

;-------------------------------------------------------------------------------
;       Specify interrupt masking
;-------------------------------------------------------------------------------
        MOVW    MK0,#0FFFFH
        MOVW    MK1,#0FFFFH                     ;Masks all interrupts

;-------------------------------------------------------------------------------
;       Enable interrupts
;-------------------------------------------------------------------------------
```

```
        EI

;********************************************************************************
;
;
;       Main processing
;
;
;********************************************************************************
MAIN_LOOP:
        ;*****************************************************;
        ;                                                     ;
        ;   Processing to transmit the measured temperature  ;
        ;                                                     ;
        ;*****************************************************;
        ;------------------------------------------;
        ;         Timing creation processing      ;
        ;------------------------------------------;
LMAIN100:
        BF      TMHIF2,$LMAIN500        ;Have 100 ms elapsed? → NO
        CLR1    TMHIF2                  ;Clears interrupt requests.
        DEC     R1SECCNT                ;Updates the 1-second counter.
        BNZ     $LMAIN500               ;Has 1 second elapsed? → NO
        MOV     R1SECCNT,#C1SEC         ;Initializes the 1-second counter.

        ;------------------------------------------;
        ;   Temperature measurement processing    ;
        ;------------------------------------------;
        MOV     B,#0            ;Specifies an argument (pulse width measurement mode).
        CALL    !SGETPULSE      ;Measures the discharge pulse width of the fixed resistor for
calibration.
        MOVW    RCALBCNT,AX     ;Acquires the measured pulse width.

        MOV     B,#1            ;Specifies an argument (pulse width measurement mode).
        CALL    !SGETPULSE      ;Measures the discharge pulse width of the thermistor.
        MOVW    RTHERMCNT,AX    ;Acquires the measured pulse width.

LMAIN400:
        CALL    !SGETHEAT       ;Calculates the resistance from the measured pulse width and
acquires the temperature.

        ;-----------------------------------------------;
        ;    Creation and transmission of UART6 data   ;
        ;-----------------------------------------------;
        CALL    !SUART6TX

LMAIN500:
        ;*******************************************;
        ;                                          ;
        ;    Different types of main processing    ;
        ;                                          ;
        ;******************************************;

        ;Any other main processing is performed here.

        BR      MAIN_LOOP

;******************************************************************************
;
;    Measurement of the capacitor discharge time (measurement of the TI000 pulse width)
;
;-------------------------------------------------------------------------------
;        [  IN  ] B: Pulse width measurement mode (0: The discharge pulse width of the fixed
resistor for calibration is measured.
;                                                    1: The discharge pulse width of the
thermistor is measured.)
;       [  OUT ] AX: Measured discharge pulse width
;                ROVFCNT: Number of times TM00 overflows
;
;       The capacitor discharge time is measured by determining the pulse width by using TI000.
;       Whether to measure the discharge pulse width of a fixed resistor for calibration
;       or a thermistor is specified by using an argument.
;       The measured discharge pulse width is returned.
;       If TM00 overflows while measuring the pulse width,
;       the number of overflows is set to the appropriate counter.
;******************************************************************************
SGETPULSE:
        MOV     A,B                     ;Acquires the pulse width measurement mode.
        MOV     ROVFCNT,#0              ;Clears the counter that counts overflows.

        ;============    Charge the capacitor    ============
        SET1    P3.3
        CLR1    PM3.3                   ;Starts charging the capacitor.
        ;Waits 2 ms for the capacitor to charge.
        MOV     B,#93           ;[4clk]                  ↑
JGETP100:                       ;                        |
        MOV     C,#27           ;[4clk]↑                 |4 + (166 + 6)*93 = 16000clk
JGETP101:                       ;      |4+6*27 = 166clk  |16000 * 0.125[μs] = 2000[μs]
        DBNZ    C,$JGETP101     ;[6clk]↓                 |
        DBNZ    B,$JGETP100     ;[6clk]                  ↓

        SET1    PM3.3                   ;Uses P33 as TI000.
        CLR1    TMIF010                 ;Clears interrupt requests.

        ;===========   Start discharging the capacitor   ===========
        MOV     B,A                     ;Saves the pulse width measurement mode.
        CMP     A,#0                    ;Has the discharge pulse width of the fixed resistor
for calibration been measured?
        BNZ     $JGETP200               ; → NO: The discharge pulse width of the thermistor is
measured.

        CLR1    P3.1                    ;Prepares to discharge. ;Starts discharging when P31 is
set to low-level output.
        MOV     TMC00,#08H              ;Starts measuring the pulse width by using 16-bit
timer/event counter 00.
        CLR1    PM3.1                   ;Starts discharging.
        BR      JGETP300
JGETP200:
        CLR1    P3.2                    ;Prepares to discharge. ;Starts discharging when P32 is
set to low-level output.
        MOV     TMC00,#08H              ;Starts measuring the pulse width by using 16-bit
timer/event counter 00.
        CLR1    PM3.2                   ;Starts discharging.

JGETP300:
        ;==========  Wait for the capacitor to discharge  ==========
        BT      TMIF010,$JGETP500       ;Has the capacitor discharged? → YES

        BF      OVF00,$JGETP400         ;Is TM00 overflow detected? → NO
        CLR1    OVF00                   ;Clears the TM00 overflow flag.
        INC     ROVFCNT                 ;Updates the number of overflows.
        CMP     ROVFCNT,#2              ;Have at least 2 overflows occurred?
        BZ      $JGETP500               ; → YES: A temperature measurement error occurs and
pulse width measurement is suspended.
JGETP400:
        BR      JGETP300                ;The wait for the capacitor to discharge continues.

JGETP500:
        CLR1    TMIF010                 ;Clears interrupt requests.

        ;===========   Finish discharging the capacitor   ===========
        MOVW    AX,CR010                ;Acquires the measured pulse width.
        DEC     B
        BZ      $JGETP700               ;Has the discharge pulse width of the fixed resistor
for calibration been measured?
                                        ; → NO: The discharge pulse width of the thermistor is
measured.
        SET1    PM3.1                   ;Sets the port used to discharge the capacitor when
using the fixed resistor for calibration back to input.
        CMP     ROVFCNT,#0              ;Has an overflow occurred when measuring the pulse
width?
        BZ      $JGETP800               ; → NO: Returns the measured pulse width.
        MOVW    AX,#0                   ;Returns the value as an error.
        BR      JGETP800
JGETP700:
        SET1    PM3.2                   ;Sets the port used to discharge the capacitor when
using a thermistor back to input.

JGETP800:
        MOV     TMC00,#0                ;Stops 16-bit timer/event counter 00.
        CLR1    P3.3
        CLR1    PM3.3                   ;Sets TI000 back to low-level output.

        RET

;******************************************************************************
;
;       Temperature acquisition processing
;
;-------------------------------------------------------------------------------
;       [  IN  ] RCALBCNT: Discharge pulse width of the fixed resistor for calibration
;                RTHERMCNT: Discharge pulse width of the thermistor
;                ROVFCNT: Number of times TM00 overflows (when measuring the discharge pulse
width of the thermistor)
;       [  OUT ] RHEAT: Temperature (BCD)
;
;       The resistance is calculated from the measured pulse width and
;       the temperature is acquired from the temperature conversion table.
;
;       The resistance is calculated from the pulse width by using the following equation:
;
;                      Rc × (CNTth + number of overflows x 10000H)
;               Rth = ----------------------------------------------
;                                        CNTc
;
;               Rth: Thermistor resistance [100 Ω]
;               Rc: Resistance of the fixed resistor for calibration = 330 [100 Ω]
;               CNTth: Discharge pulse width of the thermistor
;               CNTc: Discharge pulse width of the fixed resistor for calibration
;
;
;     ◎The value relative to the Rth measurement range is calculated by using the equation
below,
;         and the temperature is acquired from the temperature conversion table by using that
value as the offset.
;
;               Rrel = Rth - Rmin
;
;               Rrel: Value relative to the Rth measurement range [100 Ω]
;               Rmin: Minimum resistance in the measurement range = 245 [100 Ω]
;******************************************************************************
SGETHEAT:
        CMP     RCALBCNT,#0             ; Did an error occur while measuring the discharge
```

```
pulse width of the fixed resistor for calibration?
        BZ      $JGETH800               ; → YES: The resistance cannot be calculated.  The
resistance is not calculated.

        CMP     ROVFCNT,#2              ; Did at least two overflows occur while measuring the
pulse width?
        BZ      $JGETH800               ; → YES: The resistance is already outside the
measurement range.  The resistance is not calculated.

        ;---------------------------------;
        ; Calculate the resistance from the pulse width ;
        ;---------------------------------;
        MOVW    RTEMP32,#0              ;Saves 0 to the lower 16 bits of the variable used for
calculation.
        MOVW    AX,RTHERMCNT
        MOVW    (RTEMP32+2),AX          ;Saves CNTth to the higher 16 bits of the variable used
for calculation.
        MOVW    AX,#330
        MOVW    RTEMP16A,AX             ;Saves Rc (330) [100 Ω] to the variable used for
calculation.
        CALL    !SMULT16                ;Calculates (Rc x CNTth).

        ;Adds (Rc x number of overflows x 10000H) to the result of (Rc x CNTth).
        CMP     ROVFCNT,#0              ;Has an overflow occurred?
        BZ      $JGETH300               ; → NO: Calculating the resistance continues without
adding 10000H.
        MOV     A,ROVFCNT
        MOV     B,A                     ;Sets the number of overflows to the counter.
        MOVW    AX,(RTEMP32+2)          ;Adds Rc to the higher 16 bits of the result of (Rc x
CNTth).
JGETH200:
        ADDW    AX,#330                 ;Adds Rc.
        BC      $JGETH800               ;Has the result of addition overflowed? → YES: The
temperature is outside the measurement range.
        DBNZ    B,$JGETH200             ;Has Rc been added for the number of overflows? → NO
        MOVW    (RTEMP32+2),AX

JGETH300:
        MOVW    AX,RCALBCNT
        MOVW    RTEMP16A,AX             ;Specifies the discharge pulse width of the fixed
resistor for calibration as the divisor.
        CALL    !SDIV32                 ;Calculates (Rc x (CNTth + number of overflows x
10000H)/CNTc.

        ;-----------------------------------------------------------------;
        ; Determine whether the thermistor resistance is within the measurement range (24.5 kΩ
to 37.0 kΩ);
        ;-----------------------------------------------------------------;
        MOVW    AX,(RTEMP32+2)          ;Acquires the higher 16 bits of the calculated
resistance.
        CMPW    AX,#0000H               ;Compares the higher 16 bits with 0 (based on 370 =
172H).
        BNZ     $JGETH800               ;If the higher 16 bits are at least 1, the temperature
is identified as an error, because the resistance is outside the measurement range.
JGETH400:
        MOVW    AX,RTEMP32              ;Acquires the lower 16 bits of the calculated
resistance.
        CMPW    AX,#371                 ;Is the calculated resistance 37.0 kΩ or less?
        BNC     $JGETH800               ; → NO: The temperature is identified as an error.
        CMPW    AX,#245                 ;Is the calculated resistance at least 24.5 kΩ?
        BC      $JGETH800               ; → NO: The temperature is identified as an error.

        ;------------------------;
        ; Convert the resistance to a temperature   ;
        ;------------------------;
JGETH500:       ;Calculates the value relative to the Rth measurement range.
        MOVW    AX,RTEMP32
        SUBW    AX,#245                 ;Calculates Rrel = Rth − Rmin.
        MOV     A,X                     ;Acquires the lower 8 bits.  (If Rrel is within the
measurement range, Rrel falls within the 8 bits.)
        ADD     A,A                     ;Doubles Rrel and
        MOV     B,A                     ;acquires the offset in the temperature conversion
table.
        MOVW    HL,#TR2HEAT             ;Sets the address in the temperature conversion table
to HL.
        MOV     A,[HL+B]                ;Acquires the temperature (lower 8 bits).
        MOV     X,A
        INC     B
        MOV     A,[HL+B]                ;Acquires the temperature (higher 8 bits).
        MOVW    RHEAT,AX                ;Saves the temperature to a variable.
        BR      JGETH900

        ;----------------------------------------------;
        ;   Temperature setting if an error occurred while measuring the temperature   ;
        ;----------------------------------------------;
JGETH800:
        MOVW    RHEAT,#0FFFFH           ;Identifies the temperature as an error.

JGETH900:
        RET

;*******************************************************************************
;
;       Creation and transmission of UART6 data
;
;-------------------------------------------------------------------------------
```

```
;       [  IN  ] RHEAT: Temperature (BCD)
;       [  OUT ] None
;
;       The measured temperature is converted to ASCII code, set to the transmit buffer,
;       and then transmitted.
;
;       <Example of transmitted data>
;             ◎If 38.5°C was measured
;                0   1   2   3   4   5
;               | 3 | 8 | . | 5 |¥r |¥n |
;
;             ◎If an error occurred while measuring the temperature
;                0   1   2   3   4   5
;               | * | * | . | * |¥r |¥n |
;
;********************************************************************************
SUART6TX:
        ;-----------------------------------------
        ;  Processing to create UART6 transmit data in the transmit buffer
        ;-----------------------------------------
        MOVW    AX,RHEAT
        CMPW    AX,#0FFFFH              ;Has the temperature been measured?
        BZ      $JU6TX100               ; → NO

        ;The temperature is set to the transmit buffer.
        AND     A,#0FH                  ;Acquires the 10s digit.
        ADD     A,#'0'                  ;Converts the value to ASCII code.
        MOV     RTXBUF,A                ;[0]Saves the 10s digit of the temperature.

        MOV     A,X                     ;Acquires the 1s and tenth digits.
        ROR     A,1                     ;Shifts the higher 4 bits to the lower 4 bits.
        ROR     A,1
        ROR     A,1
        ROR     A,1
        AND     A,#0FH                  ;Acquires the 1s digit of the lower 4 bits.
        ADD     A,#'0'                  ;Converts the value to ASCII code.
        MOV     (RTXBUF+1),A            ;[1]Saves the 1s digit of the temperature.

        MOV     (RTXBUF+2),#'.'         ;[2]Saves the decimal point.

        MOV     A,X                     ;Acquires the 1s and tenth digits.
        AND     A,#0FH                  ;Acquires the tenth digit.
        ADD     A,#'0'                  ;Converts the value to ASCII code.
        MOV     (RTXBUF+3),A            ;[3]Saves the tenth digit of the temperature.

        BR      JU6TX200

JU6TX100:
        ; Sets **.* to the transmit buffer.
        MOV     RTXBUF,#'*'     ;[0]Saves the asterisk.
        MOV     (RTXBUF+1),#'*' ;[1]Saves the asterisk.
        MOV     (RTXBUF+2),#'.' ;[2]Saves the decimal point.
        MOV     (RTXBUF+3),#'*' ;[3]Saves the asterisk.

JU6TX200:
        MOV     (RTXBUF+4),#0DH ;[4]Saves the carriage return.
        MOV     (RTXBUF+5),#0AH ;[5]Saves the line feed.

        ;------------------------------------------
        ;          UART6 data transmission
        ;------------------------------------------
JU6TX500:
        ;=============    Start transmission    =============
        MOV     B,#6                    ;Sets up the transmission counter.
        MOVW    HL,#RTXBUF

JU6TX600:
        CLR1    STIF6                   ;Clears interrupt requests.
        MOV     A,[HL]                  ;Acquires transmit data from the transmit buffer.
        MOV     TXB6,A                  ;Transmits the data.
JU6TX700:
        BF      STIF6,$JU6TX700         ;Has 1 byte been transmitted via UART6? → NO
        CLR1    STIF6                   ;Clears interrupt requests.
        INCW    HL                      ;Updates the location of the transmit data in the
transmit buffer.
        DBNZ    B,$JU6TX600             ;Is there data not transmitted? → YES: The next data
unit is transmitted.
JU6TX800:               ;Transmission ends.
        RET

;*******************************************************************************
;
;       Function used for multiplication (16 bits * 16 bits)
;
;-------------------------------------------------------------------------------
;       [  IN  ] RTEMP16A: Multiplier
;                RTEMP32: Saves the multiplier to the higher 16 bits and 0 to the lower 16 bits.
;       [  OUT ] RTEMP32: Operation result
;*******************************************************************************
SMULT16:
        MOV     B,#16                   ; Sets up the bit counter.
JMLT120:
        CLR1    CY
        MOV     A,RTEMP32
        ROLC    A,1
        MOV     RTEMP32,A
        MOV     A,(RTEMP32+1)
        ROLC    A,1
        MOV     (RTEMP32+1),A
        MOV     A,(RTEMP32+2)
        ROLC    A,1
        MOV     (RTEMP32+2),A
        MOV     A,(RTEMP32+3)
        ROLC    A,1
        MOV     (RTEMP32+3),A           ;Left-shifts the operation result (including the
multiplicand) 1 bit.
        BNC     $JMLT220                ;MSB = 1 ? → NO

        MOV     A,RTEMP16A
        ADD     A,RTEMP32
        MOV     RTEMP32,A
        MOV     A,(RTEMP16A+1)
        ADDC    A,(RTEMP32+1)
        MOV     (RTEMP32+1),A
        MOV     A,#0
        ADDC    A,RTEMP16A
        MOV     RTEMP16A,A              ;Adds the multiplicand.
JMLT220:
        DBNZ    B,$JMLT120              ;Have 16 bits been processed? → NO

        RET
;*******************************************************************************
;
;       Function used for division (32 bits/16 bits)
;
;-------------------------------------------------------------------------------
;       [  IN  ] RTEMP16A: Divisor
;                RTEMP32: Dividend
;       [  OUT ] RTEMP32: Operation result
;                RTEMP16B: Remainder
;*******************************************************************************
SDIV32:
        MOVW    RTEMP16B,#0             ;Initializes the variable used for calculation.

        MOV     B,#32                   ;Sets up the bit counter.
JDIV120:
        CLR1    CY
        MOV     A,RTEMP16B
        ROLC    A,1
        MOV     RTEMP16B,A
        MOV     A,(RTEMP16B+1)
        ROLC    A,1
        MOV     (RTEMP16B+1),A
        MOV     A,RTEMP32
        ROLC    A,1
        MOV     RTEMP32,A
        MOV     A,(RTEMP32+1)
        ROLC    A,1
        MOV     (RTEMP32+1),A
        MOV     A,(RTEMP32+2)
        ROLC    A,1
        MOV     (RTEMP32+2),A
        MOV     A,(RTEMP32+3)
        ROLC    A,1
        MOV     (RTEMP32+3),A           ;Left-shifts the dividend 1 bit.
        MOV     A,#0
        ADDC    A,RTEMP16B
        MOV     RTEMP16B,A              ; MSB -> LSB

        SUB     A,RTEMP16A
        MOV     RTEMP16B,A
        MOV     A,(RTEMP16B+1)
        SUBC    A,(RTEMP16A+1)
        MOV     (RTEMP16B+1),A          ;RTEMP16B - RTEMP16A
        BT      RTEMP32.0,$JDIV220      ;Is borrowing possible? → YES
        BC      $JDIV180                ;RTEMP16B < RTEMP16A ? → YES

        SET1    RTEMP32.0               ;Specifies the quotient.
        BR      JDIV220

JDIV180:
        MOV     A,RTEMP16B
        ADD     A,RTEMP16A
        MOV     RTEMP16B,A
        MOV     A,(RTEMP16B+1)
        ADDC    A,(RTEMP16A+1)
        MOV     (RTEMP16B+1),A
JDIV220:
        DBNZ    B,$JDIV120              ;Have 32 bits been processed? → NO

        RET

end
```

● main.c (C language version)

```
/*******************************************************************************

        NEC Electronics     78K0/Lx3 Series

********************************************************************************
        78K0/LF3 Series        Sample program
********************************************************************************
        Temperature Measurement Program Using Port and Timer Functions
********************************************************************************
 [History]
        2008.5.--       Newly created
********************************************************************************

 [Overview]

This sample program measures the temperature by using an externally connected thermistor.
The temperature is measured every second and transmitted via the serial interface UART6.
To measure the temperature, the capacitor discharge time is separately measured by using a
fixed resistor for calibration and a thermistor, and then the thermistor resistance is
calculated by using the ratio between the discharge time and resistance.  The calculated
thermistor resistance is converted to a temperature by using the temperature conversion table.
The discharge time is measured by determining the pulse width by using 16-bit timer/event
counter 00.
The measurement range is from 32.0℃ to 42.0℃.  If a value outside this rage is measured, an
error is transmitted via the UART.
The UART used in this sample program performs only transmission.


 <Primary initial settings>

 • Setting up the vector table
 • Specifying the register bank
 • Specifying the stack pointer
 • Specifying the ROM and RAM sizes
 • Setting up the ports
 • Specifying that the CPU clock operate on the internal high-speed oscillation clock (8 MHz
(TYP.))
 • Setting up 16-bit timer/event counter 00
 • Setting up 8-bit timer H2
 • Setting up the serial interface UART6
 • Specifying interrupt masking


 <Primary main processing>

 • Processing to acquire the discharge pulse width of the fixed resistor for calibration
 • Processing to acquire the discharge pulse width of the thermistor
 • Processing to acquire the temperature
 • Processing to create UART6 transmit data and transmitting the data


 <Primary processing for measuring the discharge pulse width>

 • Charging the capacitor
 • Starting to discharge the capacitor
 • Acquiring the TM00 count value (pulse width)


 <Primary processing to acquire the temperature>

 • Calculating the thermistor resistance from the pulse width
 • Identifying the thermistor resistance as an error
 • Acquiring the temperature from the thermistor resistance


 <Primary processing to create UART6 transmit data and transmitting the data>

 • Creating data to transmit
 • Starting communication
 • Counting the transmit data
 • Setting up the transmit data


*******************************************************************************/
#pragma SFR                                     /* Enables the inclusion of special function
register (SFR) names. */
#pragma DI                                      /* Enables the inclusion of DI instructions. */
#pragma EI                                      /* Enables the inclusion of EI instructions. */
#pragma NOP                                     /* Enables the inclusion of NOP instructions.
*/

/*==============================================================================

        Function prototype declarations

==============================================================================*/
static short fn_GetPulseTime(unsigned char);    /* Discharge pulse width acquisition processing
*/
static short fn_GetHeatData(void);              /* Temperature acquisition processing */
static void fn_UART6_Tx(void);                  /* Processing to create UART6 transmit data and
transmitting the data */

/*==============================================================================
        ROM definitions

==============================================================================*/
/*------------------------------------------------------------------------------
        Table used to convert the resistance to a temperature
--------------------------------------------------------------------------------
       The temperature is referenced according to the offset based on 24.5 kΩ [100 Ω].
       BCD [0.1°C] is referenced.
------------------------------------------------------------------------------*/
const unsigned short tR2Heat[] =
{
         0x0420         /* 24.5 kΩ → 42.0 */
        ,0x0419         /* 24.6 kΩ → 41.9 */
        ,0x0418         /* 24.7 kΩ → 41.8 */
        ,0x0417         /* 24.8 kΩ → 41.7 */
        ,0x0416         /* 24.9 kΩ → 41.6 */
        ,0x0415         /* 25.0 kΩ → 41.5 */
        ,0x0414         /* 25.1 kΩ → 41.4 */
        ,0x0413         /* 25.2 kΩ → 41.3 */
        ,0x0412         /* 25.3 kΩ → 41.2 */
        ,0x0411         /* 25.4 kΩ → 41.1 */
        ,0x0410         /* 25.5 kΩ → 41.0 */
        ,0x0409         /* 25.6 kΩ → 40.9 */
        ,0x0408         /* 25.7 kΩ → 40.8 */
        ,0x0407         /* 25.8 kΩ → 40.7 */
        ,0x0406         /* 25.9 kΩ → 40.6 */
        ,0x0405         /* 26.0 kΩ → 40.5 */
        ,0x0405         /* 26.1 kΩ → 40.5 */
        ,0x0404         /* 26.2 kΩ → 40.4 */
        ,0x0403         /* 26.3 kΩ → 40.3 */
        ,0x0402         /* 26.4 kΩ → 40.2 */
        ,0x0401         /* 26.5 kΩ → 40.1 */
        ,0x0400         /* 26.6 kΩ → 40.0 */
        ,0x0399         /* 26.7 kΩ → 39.9 */
        ,0x0398         /* 26.8 kΩ → 39.8 */
        ,0x0397         /* 26.9 kΩ → 39.7 */
        ,0x0396         /* 27.0 kΩ → 39.6 */
        ,0x0395         /* 27.1 kΩ → 39.5 */
        ,0x0394         /* 27.2 kΩ → 39.4 */
        ,0x0393         /* 27.3 kΩ → 39.3 */
        ,0x0392         /* 27.4 kΩ → 39.2 */
        ,0x0392         /* 27.5 kΩ → 39.2 */
        ,0x0391         /* 27.6 kΩ → 39.1 */
        ,0x0390         /* 27.7 kΩ → 39.0 */
        ,0x0389         /* 27.8 kΩ → 38.9 */
        ,0x0388         /* 27.9 kΩ → 38.8 */
        ,0x0387         /* 28.0 kΩ → 38.7 */
        ,0x0386         /* 28.1 kΩ → 38.6 */
```

```
,0x0385         /* 28.2 kΩ → 38.5 */
,0x0384         /* 28.3 kΩ → 38.4 */
,0x0384         /* 28.4 kΩ → 38.4 */
,0x0383         /* 28.5 kΩ → 38.3 */
,0x0382         /* 28.6 kΩ → 38.2 */
,0x0381         /* 28.7 kΩ → 38.1 */
,0x0380         /* 28.8 kΩ → 38.0 */
,0x0379         /* 28.9 kΩ → 37.9 */
,0x0378         /* 29.0 kΩ → 37.8 */
,0x0378         /* 29.1 kΩ → 37.8 */
,0x0377         /* 29.2 kΩ → 37.7 */
,0x0376         /* 29.3 kΩ → 37.6 */
,0x0375         /* 29.4 kΩ → 37.5 */
,0x0374         /* 29.5 kΩ → 37.4 */
,0x0373         /* 29.6 kΩ → 37.3 */
,0x0373         /* 29.7 kΩ → 37.3 */
,0x0372         /* 29.8 kΩ → 37.2 */
,0x0371         /* 29.9 kΩ → 37.1 */
,0x0370         /* 30.0 kΩ → 37.0 */
,0x0369         /* 30.1 kΩ → 36.9 */
,0x0368         /* 30.2 kΩ → 36.8 */
,0x0368         /* 30.3 kΩ → 36.8 */
,0x0367         /* 30.4 kΩ → 36.7 */
,0x0366         /* 30.5 kΩ → 36.6 */
,0x0365         /* 30.6 kΩ → 36.5 */
,0x0365         /* 30.7 kΩ → 36.5 */
,0x0364         /* 30.8 kΩ → 36.4 */
,0x0363         /* 30.9 kΩ → 36.3 */
,0x0362         /* 31.0 kΩ → 36.2 */
,0x0361         /* 31.1 kΩ → 36.1 */
,0x0361         /* 31.2 kΩ → 36.1 */
,0x0360         /* 31.3 kΩ → 36.0 */
,0x0359         /* 31.4 kΩ → 35.9 */
,0x0358         /* 31.5 kΩ → 35.8 */
,0x0358         /* 31.6 kΩ → 35.8 */
,0x0357         /* 31.7 kΩ → 35.7 */
,0x0356         /* 31.8 kΩ → 35.6 */
,0x0355         /* 31.9 kΩ → 35.5 */
,0x0354         /* 32.0 kΩ → 35.4 */
,0x0354         /* 32.1 kΩ → 35.4 */
,0x0353         /* 32.2 kΩ → 35.3 */
,0x0352         /* 32.3 kΩ → 35.2 */
,0x0351         /* 32.4 kΩ → 35.1 */
,0x0351         /* 32.5 kΩ → 35.1 */
,0x0350         /* 32.6 kΩ → 35.0 */
,0x0349         /* 32.7 kΩ → 34.9 */
,0x0348         /* 32.8 kΩ → 34.8 */
,0x0348         /* 32.9 kΩ → 34.8 */
```

```
        ,0x0347         /* 33.0 kΩ → 34.7 */
        ,0x0346         /* 33.1 kΩ → 34.6 */
        ,0x0346         /* 33.2 kΩ → 34.6 */
        ,0x0345         /* 33.3 kΩ → 34.5 */
        ,0x0344         /* 33.4 kΩ → 34.4 */
        ,0x0343         /* 33.5 kΩ → 34.3 */
        ,0x0343         /* 33.6 kΩ → 34.3 */
        ,0x0342         /* 33.7 kΩ → 34.2 */
        ,0x0341         /* 33.8 kΩ → 34.1 */
        ,0x0341         /* 33.9 kΩ → 34.1 */
        ,0x0340         /* 34.0 kΩ → 34.0 */
        ,0x0339         /* 34.1 kΩ → 33.9 */
        ,0x0338         /* 34.2 kΩ → 33.8 */
        ,0x0338         /* 34.3 kΩ → 33.8 */
        ,0x0337         /* 34.4 kΩ → 33.7 */
        ,0x0336         /* 34.5 kΩ → 33.6 */
        ,0x0336         /* 34.6 kΩ → 33.6 */
        ,0x0335         /* 34.7 kΩ → 33.5 */
        ,0x0334         /* 34.8 kΩ → 33.4 */
        ,0x0334         /* 34.9 kΩ → 33.4 */
        ,0x0333         /* 35.0 kΩ → 33.3 */
        ,0x0332         /* 35.1 kΩ → 33.2 */
        ,0x0332         /* 35.2 kΩ → 33.2 */
        ,0x0331         /* 35.3 kΩ → 33.1 */
        ,0x0330         /* 35.4 kΩ → 33.0 */
        ,0x0330         /* 35.5 kΩ → 33.0 */
        ,0x0329         /* 35.6 kΩ → 32.9 */
        ,0x0328         /* 35.7 kΩ → 32.8 */
        ,0x0328         /* 35.8 kΩ → 32.8 */
        ,0x0327         /* 35.9 kΩ → 32.7 */
        ,0x0326         /* 36.0 kΩ → 32.6 */
        ,0x0326         /* 36.1 kΩ → 32.6 */
        ,0x0325         /* 36.2 kΩ → 32.5 */
        ,0x0324         /* 36.3 kΩ → 32.4 */
        ,0x0324         /* 36.4 kΩ → 32.4 */
        ,0x0323         /* 36.5 kΩ → 32.3 */
        ,0x0322         /* 36.6 kΩ → 32.2 */
        ,0x0322         /* 36.7 kΩ → 32.2 */
        ,0x0321         /* 36.8 kΩ → 32.1 */
        ,0x0320         /* 36.9 kΩ → 32.0 */
        ,0x0320         /* 37.0 kΩ → 32.0 */
};

/*================================================================================

        RAM definitions

================================================================================*/
```

```
unsigned char uc1secCnt;                        /* Counts 1 second by using 100 ms (TMH2) as
the base timer. */
#define         TMH2_1SEC       (1000/100)      /* Used to count 1 second. */

unsigned short ushCalibrationCnt;               /* Used to acquire the value for measuring the
TI000 pulse width for calibration. */
unsigned short ushThermistorCnt;                /* Used to acquire the value for measuring the
TI000 pulse width for the thermistor. */
unsigned char ucOVFcnt;                         /* Counter that counts the number of times TM00
overflows */
unsigned short ushHeatData;                       /* Saves the calculated temperature.  * FFFFH
is output for a measurement error. */

unsigned char ucTxBuffer[6];                    /* Transmit data buffer */
unsigned char ucTxBufferCounter;                /* Transmission counter */

/*******************************************************************************

        Initialization processing after a reset release

*******************************************************************************/
void hdwinit(void)
{
        DI();                                   /* Disables interrupts. */

/*------------------------------------------------------------------------------
        Specify the ROM and RAM sizes
--------------------------------------------------------------------------------/
        Note that the settings differ depending on the model.
        Enable the settings of the model (μPD78F0485 by default).
------------------------------------------------------------------------------*/
        /* Settings when the μPD78F0471, μPD78F0481, or μPD78F0491 is used */
        /*IMS = 0x04;                   /* Specifies the ROM size. */
        /*IXS = 0x0C;                   /* Specifies the internal expansion RAM size. */

        /* Settings when the μPD78F0472, μPD78F0482, or μPD78F0492 is used */
        /*IMS = 0xC6;                   /* Specifies the ROM size. */
        /*IXS = 0x0C;                   /* Specifies the internal expansion RAM size. */

        /* Settings when the μPD78F0473, μPD78F0483, or μPD78F0493 is used */
        /*IMS = 0xC8;                   /* Specifies the ROM size */
        /*IXS = 0x0C;                   /* Specifies the internal expansion RAM size. */

        /* Settings when the μPD78F0474, μPD78F0484, or μPD78F0494 is used */
        /*IMS = 0xCC;                   /* Specifies the ROM size */
        /*IXS = 0x0A;                   /* Specifies the internal expansion RAM size. */

        /* Settings when the μPD78F0475, μPD78F0485, or μPD78F0495 is used */
        IMS =   0xCF;                   /* Specifies the ROM size */
        IXS =   0x0A;                   /* Specifies the internal expansion RAM size. */

/*-------------------------------------------------------------------------------
        Port setup (Unused ports are set to low-level output.)
-------------------------------------------------------------------------------*/
        /* Port 1 */
        P1 =    0b00000000;             /* Sets P1 to its initial value. */
                /*++++++++--------------- P17/P16/P15/P14/P13/P12/P11/P10: Unused (0) */
        PM1 =   0b00000000;             /* Sets P1 to input or output. */
                /*++++++++--------------- PM17/PM16/PM15/PM14/PM13/PM12/PM11/PM10: Unused (0)
*/
        /* Port 2 */
        P2 =    0b00000000;             /* Sets P2 to its initial value. */
                /*++++++++--------------- P27/P26/P25/P24/P23/P22/P21/P20: Unused (0) */
        PM2 =   0b00000000;             /* Sets P2 to input or output. */
                /*++++++++--------------- PM27/PM26/PM25/PM24/PM23/PM22/PM21/PM20: Unused (0)
*/
        /* Port 3 */
        P3 =    0b00000000;             /* Sets P3 to its initial value. */
                /*|||+++++--------------- P33/P32/P31/P34/P30:Lo(0) */
                /*+++-------------------- <Fixed to 000> */
        PM3 =   0b11100110;             /* Sets P3 to input or output. */
                /*|||+|||+--------------- PM34/PM30: Unused (0) */
                /*||| |++---------------- PM32/PM31: Input (1)  Used as ports for discharging
the capacitor. */
                /*||| +------------------ PM33: Output (0)  Used as a port for charging the
capacitor.  (Used as TI000 when measuring the pulse width.) */
                /*+++-------------------- <Fixed to 111> */
        /* Port 4 */
        P4 =    0b00000000;             /* Sets P4 to its initial value. */
                /*++++++++---------------- P47/P46/P45/P44/P43/P42/P41/P40: Unused (0) */
        PM4 =   0b00000000;             /* Sets P4 to input or output. */
                /*++++++++---------------- PM47/PM46/PM45/PM44/PM43/PM42/PM41/PM40: Unused (0)
*/
        /* Port 8 */
        P8 =    0b00000000;             /* Sets P8 to its initial value. */
                /*||||++++--------------- P83/P82/P81/P80: Unused (0) */
                /*++++------------------- <Fixed to 0000> */
        PM8 =   0b11110000;             /* Sets P8 to input or output. */
                /*||||++++--------------- PM83/PM82/PM81/PM80: Unused (0) */
                /*++++------------------- <Fixed to 1111> */
        /* Port 9 */
        P9 =    0b00000000;             /* Sets P9 to its initial value. */
                /*||||++++--------------- P93/P92/P91/P90: Unused (0) */
                /*++++------------------- <Fixed to 0000> */
        PM9 =   0b11110000;             /* Sets P9 to input or output. */
                /*||||++++--------------- PM93/PM92/PM91/PM90: Unused (0) */
                /*++++------------------- <Fixed to 1111> */
        /* Port 10 */
        P10 =   0b00000000;             /* Sets P10 to its initial value. */
                /*||||++++--------------- P103/P102/P101/P100: Unused (0) */
                /*++++------------------- <Fixed to 0000> */
        PM10 =  0b11110000;             /* Sets P10 to input or output. */
                /*||||++++--------------- PM103/PM102/PM101/PM100: Unused (0) */
                /*++++------------------- <Fixed to 1111> */
        /* Port 11 */
        P11  =  0b00000100;             /* Sets P11 to its initial value. */
                /*||||+|++--------------- P113/P111/P110: Unused (0) */
                /*|||| +----------------- P112:Hi(1)*/
                /*++++------------------- <Fixed to 0000> */
        PM11 =  0b11110000;             /* Sets P11 to input or output. */
                /*||||+|++--------------- PM113/PM111/PM110: Unused (0) */
                /*|||| +----------------- PM112: Output (0)  Used as TxD6.*/
                /*++++------------------- <Fixed to 1111> */
        /* Port 12 */
        P12 =   0b00000000;             /* Sets P12 to its initial value. */
                /*|||||||+--------------- P120: Unused (0) */
                /*|||++++---------------- P124/P123/P122/P121:Read Only */
                /*+++-------------------- <Fixed to 000> */
        PM12 =  0b11111110;             /* Sets P12 to input or output. */
                /*|||||||+--------------- PM120: Unused (0) */
                /*+++++++---------------- <Fixed to 1111111> */
        /* Port 13 */
        P13 =   0b00000000;             /* Sets P13 to its initial value. */
                /*||||++++--------------- P133/P132/P131/P130: Unused (0) */
                /*++++------------------- <Fixed to 0000> */
        PM13 =  0b11110000;             /* Sets P13 to input or output. */
                /*||||++++--------------- PM133/PM132/PM131/PM130: Unused (0) */
                /*++++------------------- <Fixed to 1111> */
        /* Port 14 */
        P14 =   0b00000000;             /* Sets P14 to its initial value. */
                /*||||++++--------------- P143/P142/P141/P140: Unused (0) */
                /*++++------------------- <Fixed to 0000> */
        PM14 =  0b11110000;             /* Sets P14 to input or output. */
                /*||||++++--------------- PM143/PM142/PM141/PM140: Unused (0) */
                /*++++------------------- <Fixed to 1111> */
        /* Port 15 */
        P15 =   0b00000000;             /* Sets P15 to its initial value. */
                /*||||++++--------------- P153/P152/P151/P150: Unused (0) */
                /*++++------------------- <Fixed to 0000> */
        PM15 =  0b11110000;             /* Sets P15 to input or output. */
                /*||||++++--------------- PM153/PM152/PM151/PM150: Unused (0) */
                /*++++------------------- <Fixed to 1111> */

/*-------------------------------------------------------------------------------
```

```
        Specify the clock frequency
--------------------------------------------------------------------------------
        The clocks are specified to operate on the 8 MHz (TYP.) internal high-speed oscillation
clock.
------------------------------------------------------------------------------*/
        OSCCTL =0b00000000;             /* Clock operating mode */
                /*||||++++--------------- <Fixed to 0000> */
                /*|||+------------------- OSCSELS: Input port mode */
                /*||+-------------------- <Fixed to 0> */
                /*++--------------------- EXCLK/OSCSEL: */
                /*                        Operating mode of the high-speed system clock pin:
Input port mode */
                /*                        P121/X1,P122/X2/EXCLK: Input port */

        MOC = 0x80;                     /* Stops the X1 oscillator and disables the external
clock from the EXCLK pin. */

        MCM =   0b00000000;             /* Selects the clock to supply. */
                /*|||||+|+--------------- XSEL/MCM0: */
                /*||||| |                 Main system clock (fXP) = Internal high-speed
oscillation clock (fRH) */
                /*||||| |                 Peripheral hardware clock (fPRS) = Internal high-
speed oscillation clock (fRH) */
                /*||||| +---------------- MCS: Read Only */
                /*+++++------------------ <Fixed to 00000> */

        PCC =   0b00000000;             /* Selects the CPU clock (fCPU). */
                /*|||+|+++--------------- CSS/PCC2/PCC1/PCC0: */
                /*||| |                   CPU clock (fCPU) = fXP */
                /*||| +------------------ <Fixed to 0> */
                /*||+-------------------- CLS: Main system clock */
                /*++--------------------- <Fixed to 00> */

        RCM =   0b00000001;             /* Selects the CPU clock (fCPU). */
                /*|||||||+--------------- LSRSTOP: Stops the internal low-speed oscillator. */
                /*||||||+---------------- RSTOP: Oscillates the internal high-speed oscillator.
*/
                /*|+++++----------------- <Fixed to 00000> */
                /*+---------------------- RSTS: Read Only */

/*------------------------------------------------------------------------------
        8-bit timer H2
-------------------------------------------------------------------------------/
        8-bit timer H2 is specified as a 100 ms interval timer and is used to measure
        the temperature and as the interval for UART transmission (every second).
------------------------------------------------------------------------------*/
        TMHMD2 = 0b01100000;            /* Timer clock selection register */
                /* |||||||+-------------- TOEN2: Disables timer output. */
                /* ||||||+--------------- TOLEV2: Timer output level  Unused */
                /* ||||++---------------- TMMD21/TMMD20: Timer operation = Interval */
                /* |+++------------------ CKS22/CKS21/CKS20: Count clock fPRS/2^12 (1953.125 Hz
if fPRS is 8 MHz) */
                /* +--------------------- TMHE2: Disables timer operation.  (Enables timer
operation after the timer is set up.) */
        CMP02 = 195-1;                  /* 100ms interval: (fPRS/2^12)*0.1[sec]=195.3125 */

        TMHE2 = 1;                      /* Starts timer operation. */
        TMHIF2 = 0;                     /* Clears interrupt requests. */
        uc1secCnt = TMH2_1SEC;          /* Initializes the 1-second counter of the TMH0 base
timer. */

/*------------------------------------------------------------------------------
        16-bit timer/event counter 00
--------------------------------------------------------------------------------/
        The capacitor discharge time (pulse width) is measured to measure the temperature
sensor resistance.
------------------------------------------------------------------------------*/
        TMC00 = 0b00000000;             /* 16-bit timer mode control register 00 */
                /*|||||||+--------------- OVF00: Clears the TM00 overflow flag. */
                /*||||||+---------------- TMC001: Timer output (TO00) is inverted when TM00 and
CR000 or TM00 and CR010 match. */
                /*||||++----------------- TMC003/TMC002: Disables 16-bit timer/event counter 00.
*/
                /*++++------------------- <Fixed to 0> */
        CRC00 = 0b00000111;             /* Capture/compare control register 00 */
                /*|||||||+--------------- CRC000: Uses CR000 as a capture register. */
                /*||||||+---------------- CRC001: Triggers the capturing of CR000 in the
reverse phase of the valid edge of the TI000 pin. */
                /*|||||+----------------- CRC002: Uses CR010 as a capture register. */
                /*+++++------------------ <Fixed to 0> */
        TOC00 = 0b0000000;              /* 16-bit timer output control register 00 */
                /*|||||||+--------------- TOE00: Disables TO00 output. */
                /*||||||+---------------- TOC001: Disables the inversion of TO00 output when
CR000 and TM00 match. */
                /*||||++----------------- LVS00/LVR00: The status of the TO00 pin output does
not change. */
                /*|||+------------------- TOC004: Disables the inversion of TO00 output when
CR010 and TM00 match. */
                /*||+-------------------- OSPE00: One-shot pulse output operates as successive
pulse output. */
                /*|+--------------------- OSPT00: One-shot pulse output is not triggered by
software. */
                /*+---------------------- <Fixed to 0> */
        PRM00 = 0b00000000;             /* Prescaler mode register 00 */
                /*|||||+++--------------- PRM002/PRM001/PRM000: Setting prohibited because fPRS
= fRH. */
```

```
                /*||||+------------------ <Fixed to 0> */
                /*||++------------------- ES001/ES000: Valid edge of the TI000 pin: Falling 
edge */
                /*++--------------------- ES101/ES100: Valid edge of the TI010 pin: Falling 
edge */

/*-------------------------------------------------------------------------------------
   UART6 setup
--------------------------------------------------------------------------------------/
   UART6 is used to transmit the measurement result by using the temperature sensor.
-------------------------------------------------------------------------------------*/

        CKSR6 = 0b00000000;             /* Selects the UART6 base clock. */
                /*||||++++--------------- TPS63-60: Base clock (fXCLK6) = fPRS */
                /*++++------------------- <Fixed to 0> */

        /* Specify the value to divide the baud rate clock. */
        BRGC6 = 35;     /* Baud rate = 8*10^6[Hz]/(2 * 115200[bps]) = 34.72 */
                        /* *Fractions are rounded up to minimize errors. */
                        /* Baud rate: 115200 bps ← 114285 bps (ERR: -0.79%) */

        ASIM6 = 0b01000101;             /* Selects the UART6 operating mode. */
                /*|||||||+--------------- ISRM6: Generates an INTSR6 interrupt when a reception 
error occurs. */
                /*||||||+---------------- SL6: Number of stop bits = 1 */
                /*|||||+----------------- CL6: Data length = 8 */
                /*|||++------------------ PS61-60: No parity */
                /*||+-------------------- RXE6: Disables reception. */
                /*|+--------------------- TXE6: Enables transmission. */
                /*+---------------------- POWER6: Disables the internal operation clock. */

        ASICL6 =0b00010110;             /* Selects the start bit and inverts the TxD6 output. 
*/
                /*|||||||+--------------- TXDLV6: Normal TxD6 output */
                /*||||||+---------------- DIR6: Start bit: LSB */
                /*|||+++----------------- SBL62-60: Unused */
                /*||+-------------------- SBTT6: Unused */
                /*|+--------------------- SBRT6: Read Only */
                /*+---------------------- SBRF6: Unused */

        ISC =   0b00001000;             /* Controls switching the input. */
                /*|||||||+--------------- ISC0: Unused */
                /*||||||+---------------- ISC1: Selects the signal input from the P33/TI000 pin 
as the source of input to TI000. */
                /*|||||+----------------- ISC2: Unused */
                /*||||+------------------ ISC3: Enables input to RxD6/P113. */
                /*||++------------------- ISC5-4: TxD6=P112,RxD6=P113 */
                /*++--------------------- <Fixed to 0> */
        POWER6 =        1;              /* Enables the internal operation clock. */

/*------------------------------------------------------------------------------
        Specify interrupt masking
------------------------------------------------------------------------------*/
        MK0 = 0x0FFFF;
        MK1 = 0x0FFFF;                  /* Masks all interrupts. */

        EI();                           /* Enables interrupts */
}

/*******************************************************************************

        Main loop

*******************************************************************************/
void main(void)
{
        while(1)
        {
                /*****************************************************/
                /*                                                   */
                /*  Processing to transmit the measured temperature */
                /*                                                   */
                /*****************************************************/
                /*-----------------------------------------*/
                /*       Timing creation processing        */
                /*-----------------------------------------*/
                if(TMHIF2)
                {/* 100 ms has elapsed. */
                        TMHIF2 = 0;             /* Clears interrupt requests. */
                        uc1secCnt--;            /* Updates the 1-second counter. */
                }

                /*------------------------------------------*/
                /*    Temperature measurement processing    */
                /*------------------------------------------*/
                if(uc1secCnt == 0)
                {/* 1 second has elapsed. */
                        uc1secCnt = TMH2_1SEC; /* Clears the 1-second counter. */

                        /* Measures the discharge pulse width of the fixed resistor for
calibration. */
                        ushCalibrationCnt = fn_GetPulseTime(0);

                        /* Measures the discharge pulse width of the thermistor. */
                        ushThermistorCnt = fn_GetPulseTime(1);
                        /* Calculates the resistance from the measured pulse width and acquires
the temperature. */
                        ushHeatData = fn_GetHeatData();

                        /* Creation and transmission of UART6 data */
                        fn_UART6_Tx();
                }
                /********************************************/
                /*                                          */
                /*    Different types of main processing    */
                /*                                          */
                /********************************************/

                /* Any other main processing is performed here. */
        }
}

/*************************************************************************************

  Measurement of the capacitor discharge time (measurement of the TI000 pulse width)

--------------------------------------------------------------------------------------
        [  IN   ] mode (0: The discharge pulse width of the fixed resistor for calibration is
measured.
                        1: The discharge pulse width of the thermistor is measured.)
        [  OUT ] Measured discharge pulse width

        The capacitor discharge time is measured by determining the pulse width by using TI000.
        Whether to measure the discharge pulse width of a fixed resistor for calibration
        or a thermistor is specified by using an argument.
        The measured discharge pulse width is returned.
        If TM00 overflows while measuring the pulse width,
        the number of overflows is set to the appropriate counter.
*************************************************************************************/
static short fn_GetPulseTime(unsigned char mode)
{
        unsigned short ushRet;          /* Used to save the return value. */
        unsigned short temp;            /* Work area */

        ucOVFcnt = 0;           /* Clears the counter that counts overflows. */

        /* Charge the capacitor */
        P3.3 = 1;
        PM3.3 = 0;              /* Starts charging the capacitor. */
        for (temp = 224; temp > 0; temp--)
        {
                NOP();          /* Waits about 2 ms for the capacitor to charge. */
        }
        PM3.3 = 1;              /* Uses P33 as TI000. */
        TMIF010 = 0;            /* Clears interrupt requests. */

        /* Start discharging the capacitor */
        if(mode == 0)
        {/* Measurement of the discharge pulse width of the fixed resistor for calibration */
                P3.1 = 0;               /* Prepares to discharge. *//* Starts discharging when
P31 is set to low-level output. */
                TMC00 = 0x08;           /* Starts measuring the pulse width. */
                PM3.1 = 0;              /* Starts discharging. */
        }
        else
        {/* Measurement of the discharge pulse width of the thermistor */
                P3.2 = 0;               /* Prepares to discharge. *//* Starts discharging when
P32 is set to low-level output. */
                TMC00 = 0x08;           /* Starts measuring the pulse width. */
                PM3.2 = 0;              /* Starts discharging. */
        }

        /* Wait for the capacitor to discharge */
        while(!TMIF010)
        {
                if(OVF00)
                {/* If an overflow of TM00 has been detected */
                        OVF00 = 0;              /* Clears the TM00 overflow flag. */
                        ucOVFcnt++;             /* Updates the number of overflows. */
                        if(ucOVFcnt >= 2)       /* If at least 2 overflows have occurred */
                                break;          /* A temperature measurement error occurs and
pulse width measurement is suspended. */
                }
        }
        TMIF010 = 0;            /* Clears interrupt requests. */
        ushRet = CR010;         /* Acquires the measured pulse width. */

        /* Finish discharging the capacitor */
        if(mode == 0){          /* Sets the port used to discharge the capacitor back to input.
*/
                PM3.1 = 1;      /* If a fixed resistor for calibration is used */
                if(ucOVFcnt > 0)
                        ushRet = 0;
        }
        else
        {
                PM3.2 = 1;      /* If a thermistor is used */
        }

        TMC00 = 0x00;           /* Stops 16-bit timer/event counter 00. */
        P3.3 = 0;
        PM3.3 = 0;              /* Sets TI000 back to low-level output. */

        return ushRet;          /* Returns the pulse width. */
}

/*******************************************************************************

        Temperature acquisition processing

--------------------------------------------------------------------------------
        [  IN  ] None
        [  OUT ] Temperature (BCD)

        The resistance is calculated from the measured pulse width and
        the temperature is acquired from the temperature conversion table.

      ◎The resistance is calculated from the pulse width by using the following equation:
              (assuming that the resistance and pulse width are proportional)
                Rc : CNTc = Rth : CNTth

                          Rc × (CNTth + number of overflows x 0x10000)
                → Rth = ----------------------------------------------
                                            CNTc

                Rth: Thermistor resistance [100 Ω]
                Rc: Resistance of the fixed resistor for calibration = 330 [100 Ω]
                CNTth: Discharge pulse width of the thermistor
                CNTc: Discharge pulse width of the fixed resistor for calibration


       ◎The value relative to the Rth measurement range is calculated by using the equation
below,
          and the temperature is acquired from the temperature conversion table by using that
value as the offset.

                Rrel = Rth - Rmin

                Rrel: Value relative to the Rth measurement range [100 Ω]
                Rmin: Minimum resistance in the measurement range = 245 [100 Ω]
*******************************************************************************/
static short fn_GetHeatData(void)
{
        unsigned short ushRet;                  /* Used to save the return value. */
        unsigned long int ulTemp1;              /* RAM used for calculation */
        unsigned char ucTemp2;                  /* RAM used for calculation */

        if((ushCalibrationCnt != 0) && (ucOVFcnt < 2))
        {/* If the discharge pulse width of the fixed resistor for calibration can be measured
*/
         /* and no more than two overflows occur while measuring the discharge pulse width of
the thermistor resistance, */
         /* the resistance is calculated from the pulse width. */
                /* The measured thermistor pulse width is expanded to 32 bits by adding the
overflow portion. */
                ulTemp1 = (unsigned long)(ucOVFcnt * 0x10000) + ushThermistorCnt;
                /* The thermistor resistance is calculated. */
                ushRet = (unsigned short)((ulTemp1 * 330) / ushCalibrationCnt);

                /* Whether the thermistor resistance is within the measurement range (24.5 kΩ
to 37.0 kΩ) is determined. */
                if((ushRet <= 370)&&(ushRet >= 245))
                {/* If the resistance is within the measurement range, the temperature is
acquired from the resistance. */
                        ucTemp2 = (unsigned char)(ushRet - 245);
                        ushRet = tR2Heat[ucTemp2];
                }
                else
                {/* If the resistance is outside the measurement range, the temperature is
identified as an error. */
                        ushRet = 0xffff;
                }
        }
        else
        {/* If at least two overflows occurred while measuring the thermistor discharge pulse
width, */
        /* the resistance is already outside the measurement range. */
                ushRet = 0xffff;                /* The temperature is identified as an error.
*/
        }

        return ushRet;          /* Returns the temperature. */
}

/********************************************************************************

        Creation and transmission of UART6 data

--------------------------------------------------------------------------------
        [  IN  ] None
        [  OUT ] None

       The measured temperature is converted to ASCII code, set to the transmit buffer,
       and then transmitted.
```

```
        <Example of transmitted data>
               ◎If 38.5°C was measured
```

0	1	2	3	4	5
3	8	.	5	¥r	¥n

```
               ◎If an error occurred while measuring the temperature
```

0	1	2	3	4	5
*	*	.	*	¥r	¥n

```
********************************************************************************/
static void fn_UART6_Tx(void)
{
        /*******************************************/
        /*                                         */
        /*     Creation of UART6 transmit data     */
        /*                                         */
        /*******************************************/
        if(ushHeatData != 0xFFFF)
        {/* If the temperature has been measured, the temperature is set to the transmit buffer.
*/
                ucTxBuffer[0] = (unsigned char)(((ushHeatData >> 8) & 0x000f) + '0');   /* [0]
10s digit of the temperature (which is converted to ASCII code) */
                ucTxBuffer[1] = (unsigned char)(((ushHeatData >> 4) & 0x000f) + '0');   /* [1]
1s digit of the temperature (which is converted to ASCII code) */
                ucTxBuffer[2] = '.';                                                    /* [2]
Decimal point */
                ucTxBuffer[3] = (unsigned char)((ushHeatData & 0x000f) + '0');          /* [3]
Tenth digit of the temperature (which is converted to ASCII code) */
        }
        else
        {/* If a measurement error occurs, **.* is set to the transmit buffer. */
                ucTxBuffer[0] = '*';    /* [0]Saves the asterisk. */
                ucTxBuffer[1] = '*';    /* [1]Saves the asterisk. */
                ucTxBuffer[2] = '.';    /* [2]Saves the decimal point. */
                ucTxBuffer[3] = '*';    /* [3]Saves the asterisk. */
        }
        ucTxBuffer[4] = '¥r';           /* [4]Carriage return */
        ucTxBuffer[5] = '¥n';           /* [5]Line feed */

        /*******************************************/
        /*                                         */
        /*         UART6 data transmission         */
        /*                                         */
        /*******************************************/
        for(ucTxBufferCounter = 0; ucTxBufferCounter < sizeof(ucTxBuffer); ucTxBufferCounter++)
        {/* Transmission continues until all data has been transmitted. */
                STIF6 = 0;                              /* Clears interrupt requests. */
                TXB6 = ucTxBuffer[ucTxBufferCounter];   /* Transmits the data. */
                while(!STIF6)/* The system waits until 1 byte has been transmitted via UART6.
*/
                        NOP();
        }
}
```

APPENDIX B REVISION HISTORY

Edition	Date Published	Page	Revision
1st edition	July 2009	–	–

For further information,
please contact:

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
http://www.necel.com/

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
http://www.am.necel.com/

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
http://www.eu.necel.com/

Hanover Office
Podbielskistrasse 166 B
30177 Hannover
Tel: 0 511 33 40 2-0

Munich Office
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

Stuttgart Office
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
Tel: 010-8235-1155
http://www.cn.necel.com/

Shanghai Branch
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai, P.R.China P.C:200120
Tel:021-5888-5400
http://www.cn.necel.com/

Shenzhen Branch
Unit 01, 39/F, Excellence Times Square Building,
No. 4068 Yi Tian Road, Futian District, Shenzhen,
P.R.China P.C:518048
Tel:0755-8282-9800
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
Unit 1601-1613, 16/F., Tower 2, Grand Century Place,
193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600
http://www.tw.necel.com/

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737
http://www.kr.necel.com/