

To our customers,

---

## Old Company Name in Catalogs and Other Documents

---

On April 1<sup>st</sup>, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1<sup>st</sup>, 2010  
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

## 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 (Real-Time Counter)

### Continuing to Operate Real-Time Counter in Low-Voltage Status

---

This document summarizes the operation of the sample program and describes how to use it. This sample program continues to operate the real-time counter in a low-voltage status by using the charge in the super capacitor between the power supply and ground, even if the power is turned off for an application (such as when the battery is disconnected). The sample program transmits the counter values, which are the results of having continued to operate the real-time counter, by using UART communication after the application returns from the low-voltage status to the normal-voltage status (such as after the battery is replaced). The purpose of the sample program is to continue to operate the counter without setting up the watch data again after the application returns from the low-voltage status to the normal-voltage status.

#### Target devices

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

#### CONTENTS

CHAPTER 1 OVERVIEW.....	3
CHAPTER 2 CIRCUIT DIAGRAM.....	6
2.1 Circuit Diagram.....	6
2.2 Peripheral Hardware.....	7
CHAPTER 3 SOFTWARE.....	8
3.1 Included Files.....	8
3.2 Used Internal Peripheral Functions.....	9
3.3 Initial Settings and Operational Overview.....	10
3.4 UART Data Transmission Format.....	12
3.5 Flowcharts.....	13
CHAPTER 4 SETUP.....	16
4.1 Initial Settings of Peripheral Functions.....	16
4.2 Main Processing.....	28
4.3 Low-Voltage Detection Interrupt Servicing (Using INTLVI).....	31
4.4 UART Transmission Processing.....	33
CHAPTER 5 RELATED DOCUMENTS.....	38
APPENDIX A PROGRAM LIST.....	39
APPENDIX B REVISION HISTORY.....	68

- **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

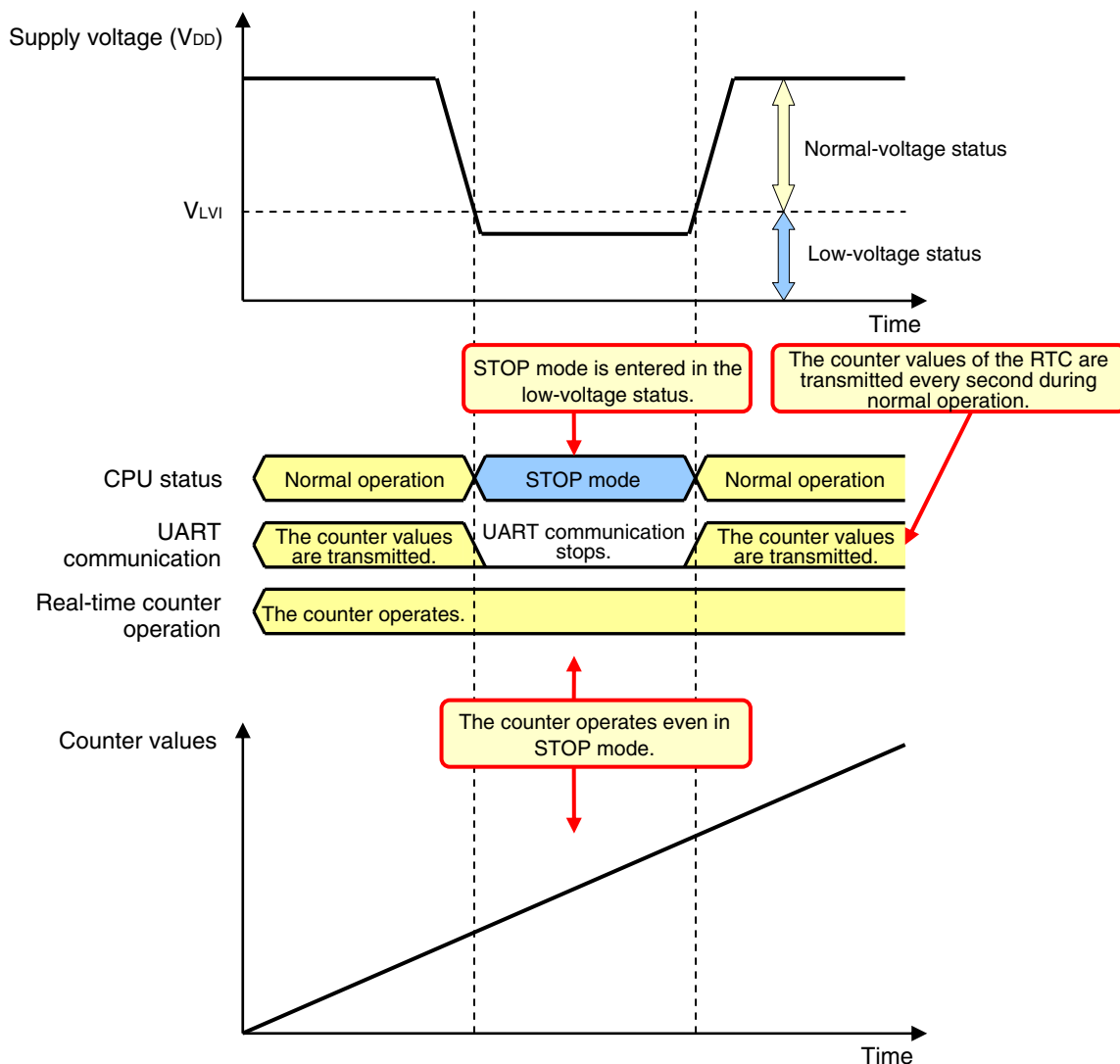
## CHAPTER 1 OVERVIEW

This sample program continues to operate the real-time counter in a low-voltage status by using the charge in the super capacitor between the power supply and ground, even if the power is turned off for an application (such as when the battery is disconnected). The sample program transmits the counter values, which are the results of having continued to operate the real-time counter, by using UART communication after the application returns from the low-voltage status to the normal-voltage status (such as after the battery is replaced). The purpose of the sample program is to continue to operate the counter without setting up the watch data again after the application returns from the low-voltage status to the normal-voltage status.

During the main processing, the counter values of the real-time counter (the year, month, day, day of the week, hour, minute, and second) are read and UART transmission processing is called.

The low-voltage status is detected by the low-voltage detector. If the supply voltage ( $V_{DD}$ ) falls below the detection voltage ( $V_{LVI}$ ) or exceeds  $V_{LVI}$ , an interrupt signal (INTLVI) is generated. If the low-voltage status is detected, STOP mode is entered by interrupt servicing. Interrupt servicing ends after returning from the low-voltage status.

During UART transmission processing, the counter values of the real-time counter (the year, month, day, day of the week, hour, minute, and second) are transmitted via the serial interface UART6.



**(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 the low-voltage detector so that it generates an interrupt signal when the supply voltage ( $V_{DD}$ ) level is detected
- Setting up the serial interface UART6 to use for data transmission
- Setting up the real-time counter to generate an interrupt at fixed periods and initializing the counter values
- Unmasking INTLVI interrupt
- Enabling interrupts

**(2) Main processing**

The processing for reading the counter values of the real-time counter (the year, month, day, day of the week, hour, minute, and second) and UART transmission processing are called.

The timing to perform the main processing is created by using an interrupt generated at fixed periods of one second by the real-time counter.

During the processing for reading the counter values of the real-time counter, the wait control for the real-time counter is set to counter value read or write mode, the counter is stopped, and the counter values (the year, month, day, day of the week, hour, minute, and second) are read. Next, the wait control for the real-time counter is set to counter operation and the counter restarts.

The count registers stop counting in the counter value read or write mode. Therefore, if STOP mode is entered in the counter value read or write mode, counting cannot be performed in STOP mode. To prevent STOP mode from being entered while the counter is stopped, interrupts are disabled while reading or writing the counter values of the real-time counter.

**(3) UART transmission processing**

During UART transmission processing, the counter values read from the real-time counter (the year, month, day, day of the week, hour, minute, and second) are converted to ASCII code and then transmitted via the serial interface UART6.

If the low-voltage status is detected before transmission, the serial interface UART6 restarts.

If the low-voltage status is detected during transmission, transmission is suspended.

For details about the UART communication settings and the transmitted data, see **3.4 UART Data Transmission Format**.

**(4) INTLVI interrupt servicing (low-voltage detection interrupt)**

An interrupt is serviced if the low-voltage status is detected.

An interrupt signal (INTLVI) is generated if the supply voltage ( $V_{DD}$ ) falls below the detection voltage ( $V_{LVI}$ ) when the voltage decreases or if  $V_{DD}$  becomes at least  $V_{LVI}$  when the voltage increases.  $V_{LVI}$  is set to  $1.93\text{ V} \pm 0.1\text{ V}$ .

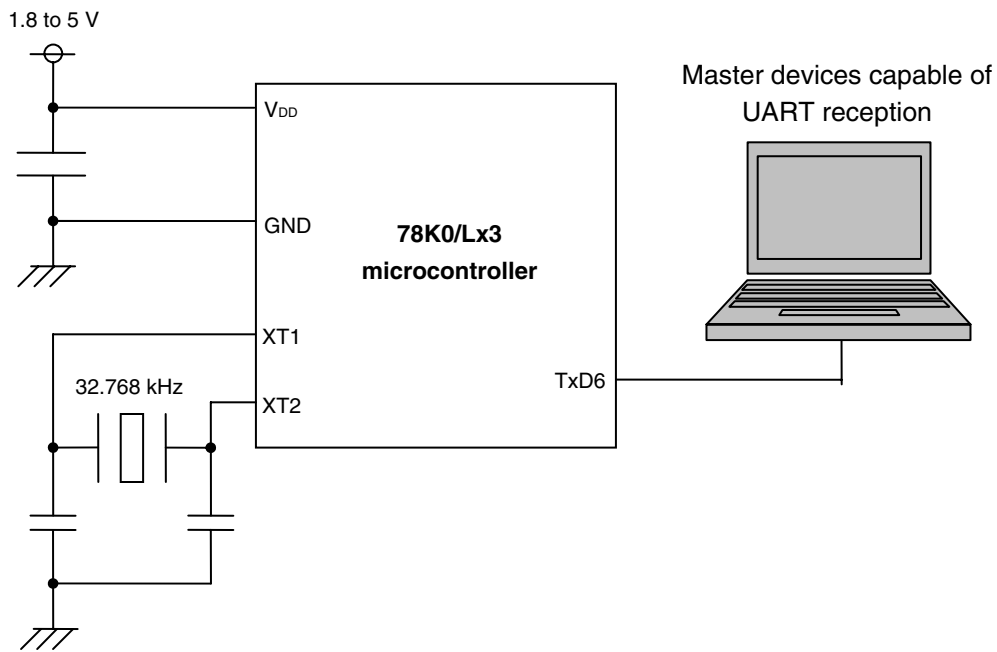
STOP mode is entered if  $V_{DD}$  falls below  $V_{LVI}$  and an INTLVI signal is generated. Interrupt servicing ends if  $V_{DD}$  becomes at least  $V_{LVI}$  and an INTLVI signal is generated. When entering STOP mode, whether the counter value read or write mode is entered is checked so that the real-time counter can operate in STOP mode.

## CHAPTER 2 CIRCUIT DIAGRAM

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

### 2.1 Circuit Diagram

The circuit diagram is shown below.



- Cautions**
1. Connect the AV<sub>REF</sub> pin directly to V<sub>DD</sub> (1.8 to 5 V supply).
  2. Connect the AV<sub>SS</sub> pin directly to GND.
  3. Leave all pins in the circuit diagram and all unused pin except the AV<sub>REF</sub> and AV<sub>SS</sub> pins open (unconnected), because they are used as output ports.
  4. Connect the TxD6 pin to a device capable of UART reception.



## 2.2 Peripheral Hardware

The peripheral hardware is described below.

**(1) Subsystem clock**

Connect a 32.768 kHz crystal resonator to the XT1 and XT2 pins.

**(2) UART communication device (TxD6)**

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

**(3) Super capacitor**

Connect a super capacitor between the  $V_{DD}$  and GND pins.



- Cautions**
1. **Connect the  $AV_{REF}$  pin directly to  $V_{DD}$  (1.8 to 5 V supply).**
  2. **Connect the  $AV_{SS}$  pin directly to GND.**

## CHAPTER 3 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.

### 3.1 Included Files

The compressed file to be downloaded includes the following files.

File Name	Description	Included Compressed Files (*.zip)	
			
main.asm (assembly language version) ----- main.c (C language version)	Source files for hardware initialization processing of the microcontroller, main processing, UART transmission processing, and low-voltage detection interrupt servicing	● 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_LVI_RTC.prw	Workspace file for the integrated development environment PM plus		●
78K0Lx3_LVI_RTC.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**



: Includes only source files.



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

## 3.2 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.
- Real-time counter  
This counter is used to count the year, month, day of the week, day, hour, minute, and second and generate interrupts at fixed periods of one second.
- Low-voltage detector  
This detector is used to detect the low-voltage status.
- Serial interface UART6  
This interface is used to transmit the counter values of the real-time counter.

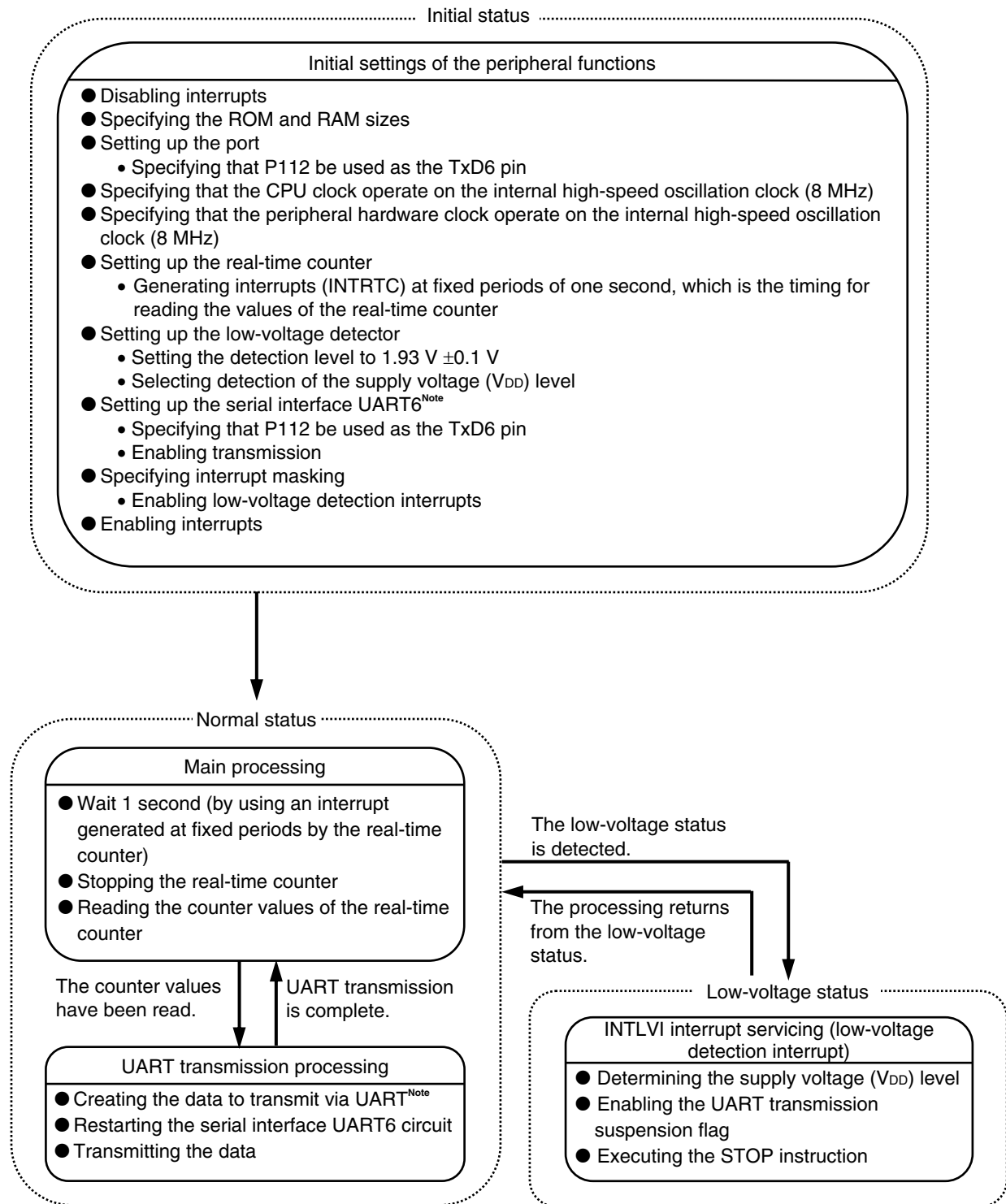
### 3.3 Initial Settings and Operational Overview

In the sample program, the ports, real-time counter, low-voltage detector, and the serial interface UART6 are set up and the clock frequency is selected as part of the initial settings for the peripheral functions. Wait one second after the initial settings for the peripheral functions are set up. The wait control of the real-time counter is set to counter value read or write mode, the counter stops, and the counter values are read. Next, the wait control of the real-time counter is set to counting, the counter starts, the read counter values are transmitted via UART communication, and there is another one-second wait. The counter values (the year, month, day of the week, day, hour, minute, and second) are transmitted via UART communication<sup>Note</sup>.

If the supply voltage ( $V_{DD}$ ) falls below 1.93 V, STOP mode is entered by interrupt servicing. If the supply voltage ( $V_{DD}$ ) becomes at least 1.93 V, the processing returns from STOP mode and interrupt servicing ends. If the low-voltage status is detected before transmission during UART communication, the serial interface UART6 circuit restarts.

For details, see the status transition diagram below.

**Note** For details about the UART communication settings and the transmitted data, see **3.4 UART Data Transmission Format**.



**Note** For details about the UART communication settings and the transmitted data, see **3.4 UART Data Transmission Format**.

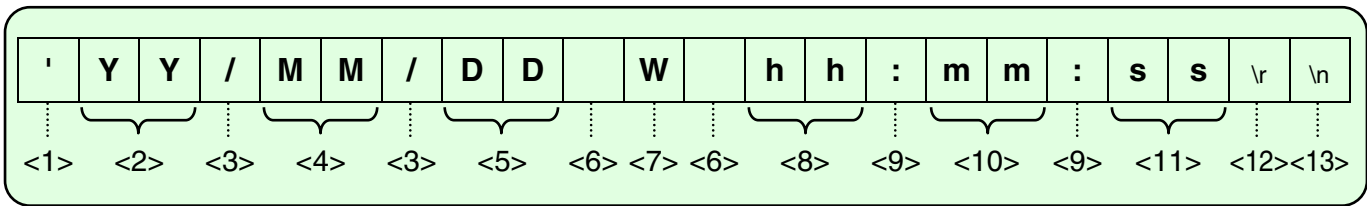
### 3.4 UART Data Transmission Format

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 every second. 22 bytes of data is transmitted per transmission. The counter values read from the real-time counter (the year, month, day of the week, day, hour, minute, and second) are converted to ASCII code and then transmitted. Figure 3-1 shows the data transmission format.

Figure 3-1. UART Data Transmission Format



- <1> Quotation mark representing the abbreviated digits of the year
- <2> Year counter value represented by the lower two digits of the year in BCD (binary-coded decimal) format
- <3> Forward slashes delimiting the year, month, and day
- <4> Month counter value represented by two digits in BCD format
- <5> Day counter value represented by two digits in BCD format
- <6> Spaces delimiting the date, day of the week, and time
- <7> Day-of-the-week counter value. The counter values and days of the week correspond as follows.

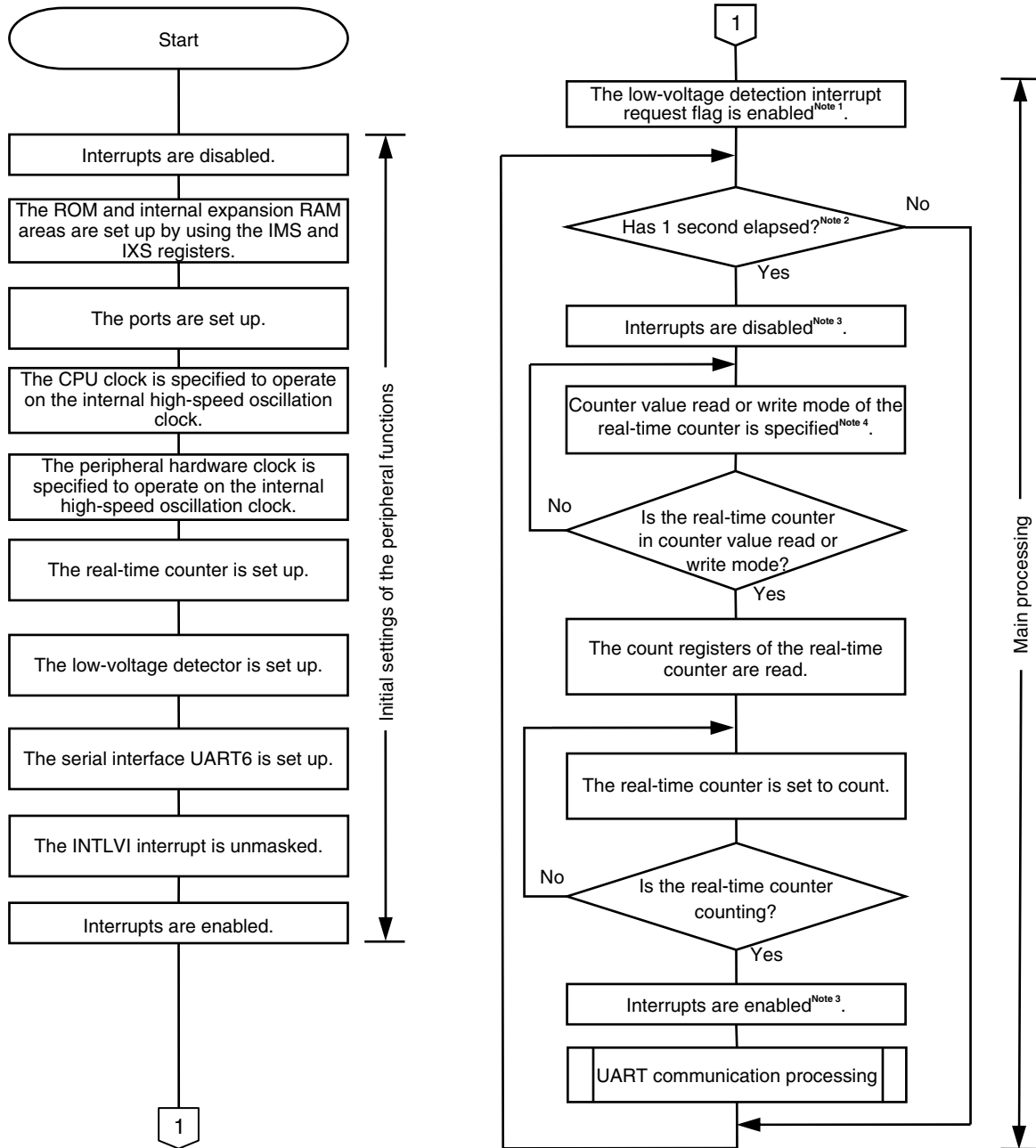
Day of the week	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Counter value	0	1	2	3	4	5	6

- <8> Hour counter value displayed in 24-hour time and represented by two digits in BCD format
- <9> Colons delimiting the hour, minute, and second
- <10> Minute counter value represented by two digits in BCD format
- <11> Second counter value represented by two digits in BCD format
- <12> Return code
- <13> Line feed code

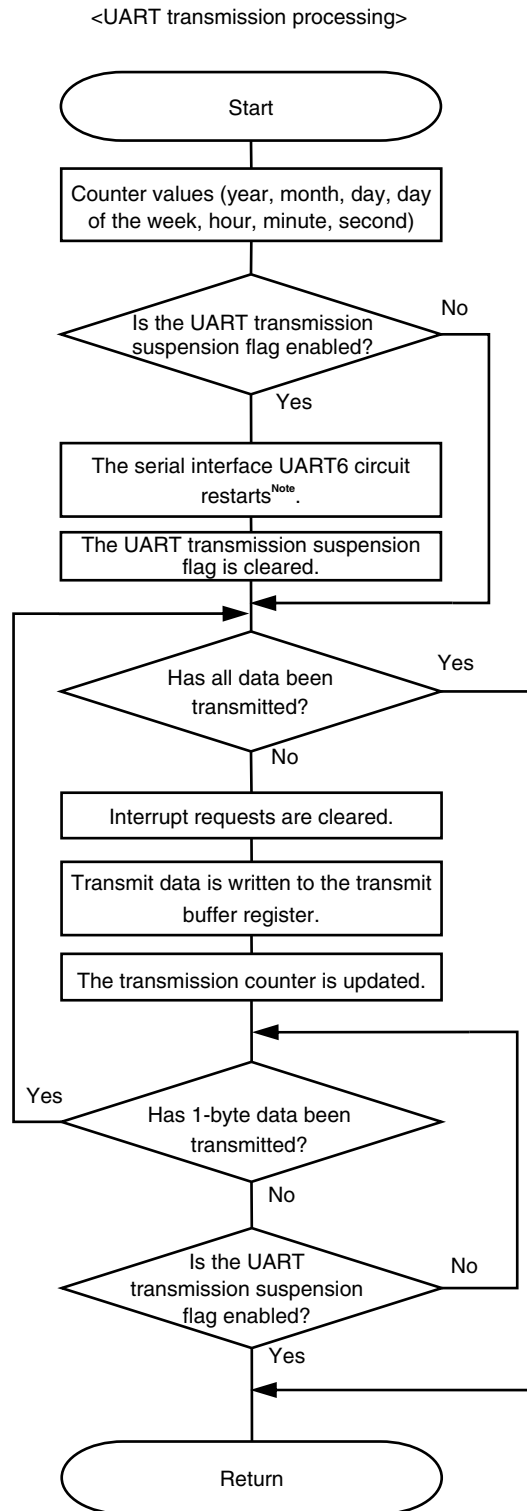
### 3.5 Flowcharts

The flowcharts for the sample program are shown below.

<Processing for setting up the peripheral functions>



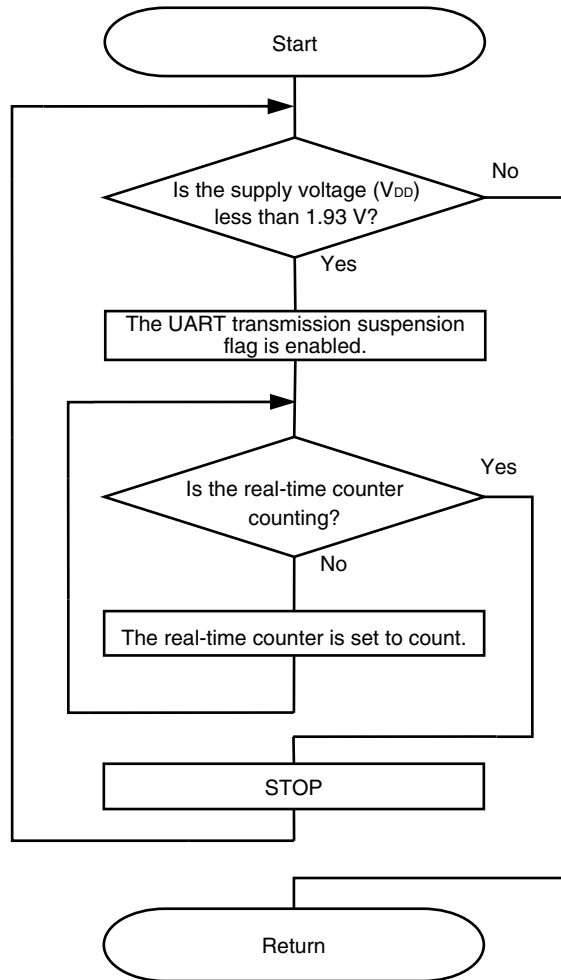
- Notes**
1. The supply voltage ( $V_{DD}$ ) level is detected by servicing a low-voltage detection interrupt after a reset starts.
  2. The system waits one second by using an interrupt (INTRTC) generated at fixed periods by the real-time counter.
  3. Entering STOP mode while the counter is stopped and the counter values of the real-time counter are not being updated is prevented.
  4. A maximum time of one clock cycle (32.768 kHz) is required for the real-time counter to stop.



**Note** If clock supply to the serial interface UART6 stops, operation after clock supply restarts is not guaranteed. Therefore, restart the circuit by setting POWER6, RXE6, and TXE6 to 0 after clock supply restarts.



&lt;Low-voltage detection interrupt servicing (using INTLVI)&gt;



## CHAPTER 4 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**.

### 4.1 Initial Settings of Peripheral Functions

#### (1) Variable definitions

The following variables are defined in assembly language:

- <1> RSEC: Area that stores the second counter value read from the real-time counter
- <2> RMIN: Area that stores the minute counter value read from the real-time counter
- <3> RHOUR: Area that stores the hour counter value read from the real-time counter
- <4> RWEEK: Area that stores the day-of-the-week counter value read from the real-time counter
- <5> RDAY: Area that stores the day counter value read from the real-time counter
- <6> RMOTH: Area that stores the month counter value read from the real-time counter
- <7> RYEAR: Area that stores the year counter value read from the real-time counter
- <8> RTXBUF: Array that stores the data to transmit by using UART communication. The counter values read from the real-time counter (the year, month, day, day of the week, hour, minute, and second) are converted to ASCII code and stored in this array.
- <9> FUARTSTOP: Flag that reports suspension of UART transmission and restarting of the serial interface UART6 circuit. This flag is enabled when the low-voltage status is detected and STOP mode is entered.

```

;=====
;
;      RAM definitions
;
;=====
DRTC  DSEG  SADDR  ;RAM related to the RTC
<1>---RSEC:   DS      1      ;Variable used to acquire the second data
<2>---RMIN:   DS      1      ;Variable used to acquire the minute data
<3>---RHOUR:  DS      1      ;Variable used to acquire the hour data
<4>---RWEEK:  DS      1      ;Variable used to acquire the day-of-the-week data
<5>---RDAY:   DS      1      ;Variable used to acquire the day data
<6>---RMOTH:  DS      1      ;Variable used to acquire the month data
<7>---RYEAR:  DS      1      ;Variable used to acquire the year data

<8>---RTXBUF:  DS     22      ;Transmit data buffer

DLVI  BSEG
<9>---FUARTSTOP DBIT      ;RAM related to INTLVI servicing
;UART transmission suspension flag
```

The following variables are defined in C language:

- <1> ucSec: Area that stores the second counter value read from the real-time counter
- <2> ucMin: Area that stores the minute counter value read from the real-time counter
- <3> ucHour: Area that stores the hour counter value read from the real-time counter
- <4> ucWeek: Area that stores the day-of-the-week counter value read from the real-time counter
- <5> ucYear: Area that stores the year counter value read from the real-time counter
- <6> ucMonth: Area that stores the month counter value read from the real-time counter
- <7> ucDay: Area that stores the day counter value read from the real-time counter
- <8> ucTxBuffer[22]: Array that stores the data to transmit by using UART communication. The counter values read from the real-time counter (the year, month, day, day of the week, hour, minute, and second) are converted to ASCII code and stored in this array.
- <9> ucTxBufferCounter: The number of data units transmitted during UART transmission processing is counted.
- <10> bUARTStop: Flag that reports suspension of UART transmission and restarting of the serial interface UART6 circuit. This flag is enabled when the low-voltage status is detected and STOP mode is entered.

```

/*=====
RAM definitions
=====*/
/* Variables used to acquire data from the real-time counter */
<1>---- unsigned char    ucSec          /* Second */
<2>----          ,ucMin      /* Minute */
<3>----          ,ucHour    /* Hour */
<4>----          ,ucWeek   /* Day of the week */
<5>----          ,ucYear   /* Year */
<6>----          ,ucMonth  /* Month */
<7>----          ,ucDay;   /* Day */

<8>---- unsigned char ucTxBuffer[22]; /* Transmit data buffer */
<9>---- unsigned char ucTxBufferCounter; /* Transmission counter */

<10>---- boolean bUARTStop; /* UART transmission suspension flag */

```

**(2) 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<sup>Note</sup> 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. 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. The operating mode of the subsystem clock pin is set to XT1 oscillation mode.
- <7> The real-time counter is set up.  
The subsystem clock (32.768 kHz) is specified as the input clock and an interrupt (INTRTC) is specified to be generated at fixed periods of one second.
  - (a) Subsystem clock oscillation stabilization wait processing. The system uses an interrupt generated at fixed periods by the real-time counter to wait one second required for subsystem clock oscillation to stabilize. Next, the real-time counter stops, each count register is set up, and counting restarts.
- <8> The low-voltage detector is set up.  
The low-voltage detector is set up so that the supply voltage ( $V_{DD}$ ) level is detected and a low-voltage detection interrupt (INTLVI) is generated. The detection level is set to  $1.93\text{ V} \pm 0.1\text{ V}$ .
  - (a) If LVION is set (1), the comparator in the LVI circuit starts operating. The system then uses software to wait until the voltage is checked by using LVIF ( $10\ \mu\text{s}$  (max.)).
- <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.  
The low-voltage detection interrupts are unmasked. If a low-voltage detection interrupt (INTLVI) occurs, the interrupt is serviced. Therefore, the low-voltage detection interrupt servicing is added in advance to the address 04H in the vector table.
- <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
;MOV  IMS,  #04H  ;Specifies the ROM size.
;MOV  IXS,  #0CH  ;Specifies the internal expansion RAM size.
<4>--
;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)
<5>--
;-----
;   Setup of port 3
;-----
MOV  P3,  #00000000B ;Sets P3 to its initial value.
;|||++++----- P34/P33/P32/P31/P30: Unused (0)
;++++----- <Fixed to 000>
MOV  PM3, #11100000B ;Sets P3 to input or output.
;|||++++----- PM34/PM33/PM32/PM31/PM30: Unused (0)
;++++----- <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>

```

&lt;5&gt;

```

;-----
; 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 111111>
;-----
; 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>

```



```

;-----;
;   Wait until the subsystem clock oscillation stabilizes   ;
;-----;
;An interrupt is generated at fixed periods of one second to wait until the subsystem
clock oscillation stabilizes.
;The real-time counter restarts after the first fixed-period interrupt request is issued.
HRST400:
BF      RTCIF,$HRST400      ;Waits until the first fixed-period interrupt occurs.
CLR1    RTCE                ;Stops the real-time counter.

;Initial count register settings: '08/06/01 (Sunday) 00:00:00
MOV     SEC,    #00H        ;Second: 00 seconds
MOV     MIN,    #00H        ;Minute: 00 minutes
MOV     HOUR,   #00H        ;Hour: 00 hours
MOV     WEEK,   #00H        ;Day of the week: Sunday
MOV     DAY,    #01H        ;Day: 1st
MOV     MONTH,  #06H        ;Month: June
MOV     YEAR,   #08H        ;Year: '08

CLR1    RTCIF      ;Clears the fixed-period interrupt request flag.
SET1    RTCE       ;Starts the real-time counter.

;-----;
;   Setup of the low-voltage detector
;-----;
MOV     LVIS,   #00001111B   ;Low-voltage detection level selection register
;|||+----- LVIS3/LVIS2/LVIS1/LVIS0: The detection level is VLVI15
(1.93 V ±0.1 V).
;++++----- <Fixed to 0000>

MOV     LVIM,   #10000000B   ;Low-voltage detection register
;||||+----- LVIF: Read only
;||||+----- LVIMD: Generates an internal interrupt signal if the
supply voltage (VDD) falls below the detection voltage (VLVI) when the voltage decreases
increases.
;||||+----- LVISEL: Detects the supply voltage (VDD) level.
;++++----- <Fixed to 0000>
;+----- LVION: Enables low-voltage detection.

;The system waits from setting (1) LVION to checking the voltage by using LVIF (10 µs (max.)).
;10 µs == 80CLK (8 MHz)
MOV     B,      #13          ;4CLK
HRST300:
DBNZ    B,      $HRST300    ;6CLK
;4 + 6 * 13 = 82 CLK

;-----;
;   Setup of the serial interface UART6
;-----;
;   The count values of the real-time counter are transmitted.
;-----;
MOV     CKSR6,  #00000000B   ;Selects the UART6 base clock.
;|||+----- TPS63-60: Base clock (fCLK6) = 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
;+----- SBT6: Unused
;+----- SBRT6: Read only
;+----- SBRF6: Unused

MOV     ISC,    #00001000B   ;Controls switching the input.
;||||+----- ISC0: Unused
;|||+----- ISC1: Uses TI000 input (normal operation).
;||+----- ISC2: Unused
;+----- ISC3: Enables input to RxD6/P113.
;||+----- ISC5-4: TxD6 = P112, RxD6 = P113
;++++----- <Fixed to 0>

SET1    POWER6                ;Enables the internal operation clock.

```



```
<10>-----  
;-----  
; Specify interrupt masking  
;-----  
MOVW MK0,#0FFFFH  
MOVW MK1,#0FFFFH ;Masks all interrupts.  
  
CLR1 LVIIIF ;Clears low-voltage detection interrupt requests.  
CLR1 LVIMK ;Unmasks low-voltage detection 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)
{
    unsigned short temp;          /* Work area */

    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. */
    /*|||++++----- P34/P33/P32/P31/P30: Unused (0) */
    /*++++----- <Fixed to 000> */
    PM3 = 0b11100000;          /* Sets P3 to input or output. */
    /*|||++++----- PM34/PM33/PM32/PM31/PM30: Unused (0) */
    /*++++----- <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 = 0b00010000; /* Clock operating mode */
/* |||+----- <Fixed to 0> */
/* |||+----- OSCSELS: Sets up the subsystem clock pin. */
/* |||+----- Operating mode of the subsystem clock pin: XT1 oscillation mode */
/* |||+----- P123/XT1,P124/XT2: Connected to a crystal or ceramic resonator */
/* |||+----- <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 0> */

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 */

```



```

/*-----
Setup of the serial interface UART6
-----
The count values of the real-time counter are transmitted.
-----*/
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: Uses TI000 input (normal operation). */
/*|||++++----- 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 = 0x0FFFFF;
MK1 = 0x0FFFFF;           /* Masks all interrupts. */

LVIIF = 0;                /* Clears low-voltage detection interrupt requests. */
LVIMK = 0;                /* Unmasks low-voltage detection interrupts. */

EI();                      /* Enable interrupts */
}

```

## 4.2 Main Processing

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

- <1> The low-voltage detection interrupt request flag is enabled to detect the supply voltage ( $V_{DD}$ ) level by servicing a low-voltage detection interrupt after a reset starts.
- <2> The interrupt generated at fixed periods of one second is checked. If a fixed-period interrupt occurs, <3> to <8> are performed.
- <3> During the processing of <5>, wait control of the real-time counter is set to counter value read or write mode. In these modes, the counter stops and the count register is not updated. Interrupts are disabled to prevent STOP mode from being entered while the counter is stopped.
- <4> Wait control of the real-time counter is set to counter value read or write mode and the system waits until either mode is entered. The maximum wait time is one subsystem clock cycle (32.768 kHz).
- <5> The counter values (the year, month, day, day of the week, hour, minute, and second) of the count register of the real-time counter are read.
- <6> The real-time counter is specified to count, and the system waits until counting starts.
- <7> Interrupts are enabled.
- <8> The UART transmission processing subroutine is called.
- <9> An NOP instruction that ensures that interrupts are serviced is executed. If the registers that control interrupts are manipulated, the acknowledgment of interrupt requests is held pending until the next instruction ends. Therefore, if the processing of <9> is not performed, operations such as the following are repeated and interrupts might not be serviced:
  - The main processing branches from <2> to LMAIN200. (Interrupt requests are held pending, because this manipulation is intended for registers that control interrupts.)
  - The main processing branches from <10> to <2>. (Interrupt requests are held pending, because this instruction follows the manipulation intended for registers that control interrupts.)
- <10> The main processing branches to <2>.

```

;*****
;
;   Main processing
;
;*****
MAIN_LOOP:
    ;The supply voltage after a reset is released is checked by servicing a low-voltage
    ;detection interrupt.
    ;If the supply voltage is less than or equal to the detection voltage, STOP mode is
    ;entered.
<1>-- SET1    LVIIIF                ;Enables low-voltage detection interrupt requests.
    ;*****
    ;
    ;   RTC count value transmission processing
    ;
    ;*****
<2>-- BF     RTCIF,$LMAIN500        ;Has 1 second elapsed? →NO
    CLR1    RTCIF                ;Clears interrupt requests.
    ;-----
    ; Acquire data from the real-time counter
    ;-----
<3>-- DI                      ;The second to year counters stop while the values of the
    ;real-time counter are read.
    ;Therefore, if STOP mode is subsequently entered, the time
    ;is not incremented.
    ;Interrupts are disabled during reading.
LMAIN200:
<4>-- SET1    RWAIT                ;Stops the second to year counters (in counter value
    ;read or write mode).
    BF     RWST,$LMAIN200        ;Waits until counter value read or write mode is
    ;entered.
    ;(Waits up to one 32.768 kHz clock cycle (30.5 μs)
    ;after RWAIT is set to 1.)
    MOV     A,SEC                ;Acquires the second data.
    MOV     RSEC,A
    MOV     A,MIN                ;Acquires the minute data.
    MOV     RMIN,A
    MOV     A,HOUR               ;Acquires the hour data.
    MOV     RHOURL,A
    MOV     A,WEEK               ;Acquires the day-of-the-week data.
    MOV     RWEEK,A
    MOV     A,DAY                ;Acquires the day data.
    MOV     RDAY,A
    MOV     A,MONTH              ;Acquires the month data.
    MOV     RMOTH,A
    MOV     A,YEAR               ;Acquires the year data.
    MOV     RYEAR,A
LMAIN300:
<6>-- CLR1    RWAIT                ;Specifies that the second to year counters operate.
    BT     RWST,$LMAIN300        ;Waits until counter value read or write mode is
    ;canceled.
<7>-- EI                      ;Finishes reading and enables interrupts.
    ;-----
    ;   Creation and transmission of UART6 data
    ;-----
<8>-- CALL    !SUART6TX
LMAIN500:
<9>-- NOP                      ;The acknowledgment of interrupt requests is held pending while a
    ;manipulation instruction is executed for registers that control
    ;interrupts.
    ;An NOP instruction is executed to prevent interrupts from being
    ;disabled.
    ;*****
    ;
    ;   Different types of main processing
    ;
    ;*****
    ; Any other main processing is performed here.
<10>-- BR     MAIN_LOOP

```

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

```

/*****
Main loop
*****/
void main(void)
{
    /* The supply voltage after a reset is released is checked by servicing a low-voltage
detection interrupt. */
    /* If the supply voltage is less than or equal to the detection voltage, STOP mode is
entered. */
    LVIIF = 1;          /* Enables low-voltage detection interrupt requests. */

    while(1){
        /*****
        /*
        /*          RTC count value transmission processing          */
        /*
        /*****
        if(RTCIF)
        { /* 1 second has elapsed. */
            RTCIF = 0;      /* Clears interrupt requests. */
            /*-----*/
            /* Acquire data from the real-time counter */
            /*-----*/
            DI(); /* The second to year counters stop while the values of the real-
time counter are read. */
            /* Therefore, if STOP mode is subsequently entered, the time is not
incremented. */
            /* Interrupts are disabled during reading. */

            do{
                RWAIT = 1;    /* Stops the second to year counters */
                                /* (in counter value read or write mode). */
            }while(!RWST); /* Waits until counter value read or write mode is
entered. */
                                /* (Waits up to one 32.768 kHz clock cycle (30.5 µs) after
RWAIT is set to 1.) */

                ucSec = SEC;      /* Acquires the second data. */
                ucMin = MIN;      /* Acquires the minute data. */
                ucHour = HOUR;    /* Acquires the hour data. */
                ucWeek = WEEK;    /* Acquires the day-of-the-week data. */
                ucDay = DAY;      /* Acquires the day data. */
                ucMonth = MONTH;  /* Acquires the month data. */
                ucYear = YEAR;    /* Acquires the year data. */

                do{
                    RWAIT = 0;    /* Specifies that the second to year counters
operate. */
                }while(RWST); /* Waits until counter value read or write mode is
canceled. */

                EI(); /* Finishes reading and enables interrupts. */

                /*-----*/
                /*      Creation and transmission of UART6 data      */
                /*-----*/
                fn_UART6_Tx();
            }

            NOP(); /* The acknowledgment of interrupt requests is held pending while
*/
                    /* a manipulation instruction is executed for registers that
control interrupts.*/
                    /* An NOP instruction is executed to prevent interrupts from being
disabled.*/

                /*****
                /*
                /*          Different types of main processing          */
                /*
                /*****

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

```



### 4.3 Low-Voltage Detection Interrupt Servicing (Using INTLVI)

The following operations are performed during the low-voltage detection interrupt servicing in assembly language:

- <1> The register bank is switched.
- <2> Whether the supply voltage ( $V_{DD}$ ) is less than or at least 1.93 V is determined. If the supply voltage is less than 1.93 V, the processing of <4> is executed. If the supply voltage is at least 1.93 V, interrupt servicing ends.
- <3> The UART suspension flag<sup>Note</sup> is enabled.
- <4> If the counter values of the real-time counter (the year, month, day, day of the week, hour, minute, and second) are being read or written, counting is specified and the system waits until counting starts.
- <5> STOP mode is entered.
- <6> If a low-voltage detection interrupt (INTLVI) occurs, the processing returns from STOP mode and the processing of <2> is performed.

**Note** This flag is enabled if the low-voltage status is detected and STOP mode is entered and reports suspension of UART transmission and restarting of the serial interface UART6 circuit.

```

;*****
;
;       Low-voltage detection completion interrupts (using INTLVI)
;
;-----
;       If VDD is at least or less than the detection voltage, an interrupt is generated.
;       The detection voltage is 1.93 V.
;       *Register bank 1 is used.
;*****
IINTLVI:
<1>--  SEL      RB1                ;Specifies the register bank by servicing an
      interrupt.

HLVI100:
<2>--  BF       LVIF,$HLVI600     ;Has VDD fallen below 1.93 V? →YES

<3>--  SET1     FUARTSTOP         ;Enables the UART suspension flag.

HLVI300:
<4>--  BF       RWST,$HLVI400    ;If the real-time counter is being read
      CLR1     RWAIT             ;Cancels reading and then enters STOP mode.
      BR       HLVI300          ;Waits until reading has been canceled.

HLVI400:
<5>--  STOP                    ;Shifts to STOP mode.
<6>--  BR       HLVI100         ;Returns from STOP mode using INTVLI.

HLVI600:
      ;STOP mode is canceled because VDD is at least 1.93 V.

HLVI900:
      RETI

```

### ● Example of setting up interrupt servicing in C language ●

Interrupt servicing is set up in C language by using a #pragma directive to add interrupt servicing to the interrupt vector table corresponding to the specified interrupt request name. How to do this is described below.

- Using a #pragma directive to add interrupt servicing to the interrupt vector table (at the start of the C source code)

```
#pragma interrupt INTLVI fn_inlvi RB1
```

Specifying the register bank:  
The register bank used for the function is specified.

Function name:  
This is the name of the function that includes the interrupt servicing.

Interrupt request name:  
This is the uppercase interrupt request name. For details, see the device user's manual.

- Declaring that the interrupt servicing is a hardware interrupt function executed by using an interrupt function modifier (in the C source code)

```
__interrupt void fn_intlvi(void)
{
    ...(Omitted)...
}
```

For details, see the **CC78K0 Compiler Package Language User's Manual**.

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

```

/*****
Low-voltage detection completion interrupts (using INTLVI)
-----
If VDD is at least or less than the detection voltage, an interrupt is generated.
The detection voltage is 1.93 V.
*****/
__interrupt void fn_intlvi(void)
{
    while(LVIF == 1)      /* Enters STOP mode if VDD < 1.93 V. */
    {
        bUARTStop = 1; /* Enables the UART6 transmission suspension flag. */

        while(RWST)     /* If the real-time counter is being read */
        {
            RWAIT = 0;  /* Cancels reading and then enters STOP mode. */
        }
        STOP();         /* Shifts to STOP mode. */
    }

    /* STOP mode is canceled if VDD is at least 1.93 V. */
}

```

## 4.4 UART Transmission Processing

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

- <1> The counter values read from the real-time counter (the year, month, day, day of the week, hour, minute, and second) are converted to ASCII code.<sup>Note</sup>
- <2> If the low-voltage status is detected, the serial interface UART6 circuit restarts.
- <3> The counter values (the year, month, day, day of the week, hour, minute, and second) converted to ASCII code are transmitted via the serial interface UART6. If the low-voltage status is detected while the system waits until one byte is transmitted, transmission is suspended.
- <4> A subroutine to convert a 1-byte BCD code to ASCII code is executed.

**Note** For details about the UART communication settings and the transmitted data, see **3.4 UART Data Transmission Format**.

```

;*****
;
;   Creation and transmission of UART6 data
;
;-----
;   [ IN ] RSEC,RMIN,RHOUR,RWEEK,RDAY,RMONTH,RYEAR: RTC count values
;   [ OUT ] None
;
;   The values read from the real-time counter are converted to
;   ASCII code, set to the transmit buffer, and then transmitted.
;   If low voltage is detected before transmission starts, UART6 is set up again.
;   If low voltage is detected while the system waits until transmission ends, UART6
transmission is suspended.
;
;   <Transmit data>
;   0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
;
;   | Y | Y | / | M | M | / | D | D |   | W |   | h | h | : | m | m | : | s | s | \r | \n
;   |---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
;
;   (Y: Year, M: Month, D: Day, W: Day-of-the-week, h: Hour, m: Minute, s: Second)
;*****
SUART6TX:
;-----
;   Processing to create UART6 transmit data in the transmit buffer
;-----
MOV     RTXBUF,#''''           ;[0]Saves the quotation mark (used to display the
abbreviated year).

MOV     A,RYEAR                ;Acquires the year data.
CALL    !SNUM2CHR              ;Converts the year data to ASCII code.
MOV     (RTXBUF+1),A           ;[1]Saves the higher bit of the year data.
MOV     A,X
MOV     (RTXBUF+2),A           ;[2]Saves the lower bit of the year data.

MOV     (RTXBUF+3),# '/'       ;[3]Saves the forward slash (used to delimit the date).

MOV     A,RMOTH                ;Acquires the month data.
CALL    !SNUM2CHR              ;Converts the month data to ASCII code.
MOV     (RTXBUF+4),A           ;[4]Saves the higher bit of the month data.
MOV     A,X
MOV     (RTXBUF+5),A           ;[5]Saves the lower bit of the month data.

MOV     (RTXBUF+6),# '/'       ;[6]Saves the forward slash (used to delimit the date).

MOV     A,RDAY                 ;Acquires the day data.
CALL    !SNUM2CHR              ;Converts the day data to ASCII code.
MOV     (RTXBUF+7),A           ;[7]Saves the higher bit of the day data.
MOV     A,X
MOV     (RTXBUF+8),A           ;[8]Saves the lower bit of the day data.

```

<1>

```

MOV     (RTXBUF+9),#' '      ;[9]Saves the space.
MOV     A,RWEEK              ;Acquires the day-of-the-week data.
CALL    !SNUM2CHR            ;Converts the day-of-the-week data to ASCII
code.
MOV     (RTXBUF+10),A        ;[10]Saves the day-of-the-week data.
MOV     (RTXBUF+11),#' '    ;[11]Saves the space.
MOV     A,RHOUR              ;Acquires the hour data.
CALL    !SNUM2CHR            ;Converts the hour data to ASCII code.
MOV     (RTXBUF+12),A       ;[12]Saves the higher bit of the hour data.
<1> MOV     A,X
MOV     (RTXBUF+13),A       ;[13]Saves the lower bit of the hour data.
MOV     (RTXBUF+14),#':'    ;[14]Saves the colon (used to delimit the time).
MOV     A,RMIN               ;Acquires the minute data.
CALL    !SNUM2CHR            ;Converts the minute data to ASCII code.
MOV     (RTXBUF+15),A       ;[15]Saves the higher bit of the minute data.
MOV     A,X
MOV     (RTXBUF+16),A       ;[16]Saves the lower bit of the minute data.
MOV     (RTXBUF+17),#':'    ;[17]Saves the colon (used to delimit the time).
MOV     A,RSEC               ;Acquires the second data.
CALL    !SNUM2CHR            ;Converts the second data to ASCII code.
MOV     (RTXBUF+18),A       ;[18]Saves the higher bit of the second data.
MOV     A,X
MOV     (RTXBUF+19),A       ;[19]Saves the lower bit of the second data.
MOV     (RTXBUF+20),#0DH    ;[20]Saves the carriage return.
MOV     (RTXBUF+21),#0AH    ;[21]Saves the line feed.

;-----
;          UART6 data transmission
;-----
BF      FUARTSTOP,$JU6TX500 ;Has low voltage been detected? →NO

;==== Set up the serial interface UART6 again ====
CLR1    POWER6              ;Disables the internal operation clock.
CLR1    TXE6                ;Disables transmission.

<2> DI                      ;Disables interrupts while setting up UART6 again.
SET1    TXE6                ;Enables transmission.
SET1    POWER6              ;Enables the internal operation clock.
CLR1    FUARTSTOP           ;Clears the UART transmission suspension flag.
EI                      ;Enable interrupts

JU6TX500:
;===== Start transmission =====
MOV     B,#22                ;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.

<3> JU6TX700:
BT      FUARTSTOP,$JU6TX800 ;Is the UART6 transmission suspension flag enabled?
→YES: Transmission ends immediately.
BF      STIF6,$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

```

&lt;4&gt;

```

;*****
;
;   Convert the BCD code to ASCII code
;
;-----
;   [ IN ] A: Numeral (BCD)
;   [ OUT ] A: Higher 4 bits of the ASCII code
;           X: Lower 4 bits of the ASCII code
;*****
SNUM2CHR:
    MOV     X,A
    AND     A,#0FH           ;Acquires the lower 4 bits.
    ADD     A,#'0'          ;Converts the lower 4 bits to ASCII code.
    XCH     A,X             ;Saves the return value and acquires the value to be
converted.

    AND     A,#0F0H         ;Acquires the higher 4 bits.
    ROL     A,1             ;Acquires the higher 4 bits by shifting them to the lower 4
bits.
    ROL     A,1
    ROL     A,1
    ROL     A,1
    ADD     A,#'0'          ;Converts the higher 4 bits to ASCII code.
    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 values read from the real-time counter are converted to
ASCII code, set to the transmit buffer, and then transmitted.
If low voltage is detected before transmission starts, UART6 is set up again.
If low voltage is detected while the system waits until transmission ends, UART6 transmission
is suspended.

<Transmit data>
  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
  '  Y  Y  /  M  M  /  D  D      W      h  h  :  m  m  :  s  s  \r  \n
  (Y: Year, M: Month, D: Day, W: Day-of-the-week, h: Hour, m: Minute, s: Second)
*****/
static void fn_UART6_Tx(void)
{
    /*****/
    /*
    /*      Processing to create UART6 transmit data      */
    /*
    /*****/
    ucTxBuffer[0] = '\'';          /* Quotation mark (used to display
the abbreviated year) */
    ucTxBuffer[1] = (( ucYear & ( 0xF0 ) ) >> 4 ) + '0'; /* Higher bits of the year data
(converted to ASCII code) */
    ucTxBuffer[2] = ( ucYear & ( 0x0F ) ) + '0';          /* Lower bits of the year data
(converted to ASCII code) */
    ucTxBuffer[3] = '/';          /* Forward slash (used to delimit the
date) */
    ucTxBuffer[4] = (( ucMonth & ( 0xF0 ) ) >> 4 ) + '0'; /* Higher bits of the month data
(converted to ASCII code) */
    ucTxBuffer[5] = ( ucMonth & ( 0x0F ) ) + '0';          /* Lower bits of the month data
(converted to ASCII code) */
    ucTxBuffer[6] = '/';          /* Forward slash (used to delimit the
date) */
    ucTxBuffer[7] = (( ucDay & ( 0xF0 ) ) >> 4 ) + '0';   /* Higher bits of the day data
(converted to ASCII code) */
    ucTxBuffer[8] = ( ucDay & ( 0x0F ) ) + '0';           /* Lower bits of the day data
(converted to ASCII code) */
    ucTxBuffer[9] = ' ';          /* Space */
    ucTxBuffer[10] = ( ucWeek & ( 0x0F ) ) + '0';         /* Day-of-the-week data */
    ucTxBuffer[11] = ' ';          /* Space */
    ucTxBuffer[12] = (( ucHour & ( 0xF0 ) ) >> 4 ) + '0'; /* Higher bits of the hour data
(converted to ASCII code) */
    ucTxBuffer[13] = ( ucHour & ( 0x0F ) ) + '0';         /* Lower bits of the hour data
(converted to ASCII code) */
    ucTxBuffer[14] = ':';          /* Colon (used to delimit the time)
*/
    ucTxBuffer[15] = (( ucMin & ( 0xF0 ) ) >> 4 ) + '0';   /* Higher bits of the minute data
(converted to ASCII code) */
    ucTxBuffer[16] = ( ucMin & ( 0x0F ) ) + '0';          /* Lower bits of the minute data
(converted to ASCII code) */
    ucTxBuffer[17] = ':';          /* Colon (used to delimit the time)
*/
    ucTxBuffer[18] = (( ucSec & ( 0xF0 ) ) >> 4 ) + '0';   /* Higher bits of the second data
(converted to ASCII code) */
    ucTxBuffer[19] = ( ucSec & ( 0x0F ) ) + '0';          /* Lower bits of the second data
(converted to ASCII code) */
    ucTxBuffer[20] = '\r';          /* Carriage return */
    ucTxBuffer[21] = '\n';          /* Line feed */
}

```

```

/*****
/*
/*          UART6 data transmission          */
/*
/*****
/*-----*/
/* Set up the serial interface UART6 again */
/*-----*/
if(bUARTStop) /* If low voltage has been detected, UART6 must be set up again. */
{
    POWER6 = 0;          /* Disables the internal operation clock. */
    TXE6 = 0;           /* Disables transmission. */

    DI();               /* Disables interrupts while setting up UART6 again. */
    TXE6 = 1;           /* Enables transmission. */
    POWER6 = 1;         /* Enables the internal operation clock. */
    bUARTStop = 0;      /* Clears the UART transmission suspension flag. */
    EI();               /* Enables interrupts. */
}

/*-----*/
/*          Start 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. */
        if(bUARTStop)
        { /* If the transmission suspension flag is enabled during transmission,
transmission is suspended. */
            ucTxBufferCounter = sizeof(ucTxBuffer);
            break;
        }
    }
}
}
}

```

## CHAPTER 5 RELATED DOCUMENTS

Document Name	English
78K0/LC3 User's Manual	<a href="#">PDF</a>
78K0/LD3 User's Manual	<a href="#">PDF</a>
78K0/LE3 User's Manual	<a href="#">PDF</a>
78K0/LF3 User's Manual	<a href="#">PDF</a>
78K/0 Series Instructions User's Manual	<a href="#">PDF</a>
RA78K0 Assembler Package User's Manual	Language <a href="#">PDF</a>
	Operation <a href="#">PDF</a>
CC78K0 C Compiler User's Manual	Language <a href="#">PDF</a>
	Operation <a href="#">PDF</a>
PM+ Project Manager User's Manual	<a href="#">PDF</a>



## 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
;*****
;       Continuing to Operate Real-Time Counter in Low-Voltage Status
;*****
; [History]
;       2008.05.--           Newly created
;*****
;
; [Overview]
;
;This sample program continues to operate the real-time counter in the
;low-voltage status.
;The values of the count register of the real-time counter are acquired every second and
;transmitted via the serial interface UART6. If the supply voltage falls below 1.93 V, STOP
;mode is entered. If the supply voltage exceeds 1.93 V, the processing returns from STOP mode.
;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 the real-time counter
; • Setting up the low-voltage detector
; • Setting up the serial interface UART6
; • Specifying interrupt masking
;
;
; <Primary main processing>
;
; • Stopping the real-time counter
```

```

; • Acquiring the values of the count register of the real-time counter
; • Specifying the real-time counter to count
; • Processing to create UART6 transmit data and transmitting the data
;
;
; <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
;
;
; <Low-voltage detection interrupts (using INTLVI)>
;
; • Specifying the real-time counter to count
; • Executing the STOP instruction
;
;
;*****

```

```

;=====
;
;      Setting up the vector table
;
;=====

```

TVCT1	CSEG	AT	000000H		
	DW	RESET_START		;(00)	RESET input, POC, LVI, WDT, TRAP
TVCT2	CSEG	AT	000004H		
	DW	IINTLVI		;(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

**APPENDIX A PROGRAM LIST**

---

```

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)  INTREERR/INTGTP/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-byte stack area.
STACKTOP:              ;The first address of the stack area is FB20H.

;=====
;
;   RAM definitions
;
;=====
DRTC      DSEG   SADDR   ;RAM related to the RTC
RSEC:      DS      1              ;Variable used to acquire the second data
RMIN:      DS      1              ;Variable used to acquire the minute data
RHOUR:     DS      1              ;Variable used to acquire the hour data
RWEED:     DS      1              ;Variable used to acquire the day-of-the-week data
RDAY:      DS      1              ;Variable used to acquire the day data
RMOTH:     DS      1              ;Variable used to acquire the month data
RYEAR:     DS      1              ;Variable used to acquire the year data

RTXBUF:    DS      22             ;Transmit data buffer

DLVI      BSEG              ;RAM related to INTLVI servicing
FUARTSTOP DBIT              ;UART transmission suspension flag

;*****
;
;
;   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,     #0000000B      ;Sets P1 to its initial value.
;+++++----- P17/P16/P15/P14/P13/P12/P11/P10: Unused (0)
        MOV     PM1,    #0000000B      ;Sets P1 to input or output.

```

APPENDIX A PROGRAM LIST

```

;+++++----- PM17/PM16/PM15/PM14/PM13/PM12/PM11/PM10: Unused
(0)
;-----
;      Setup of port 2
;-----
      MOV     P2,     #0000000B           ;Sets P2 to its initial value.
;+++++----- P27/P26/P25/P24/P23/P22/P21/P20: Unused (0)
      MOV     PM2,    #0000000B           ;Sets P2 to input or output.
;+++++----- PM27/PM26/PM25/PM24/PM23/PM22/PM21/PM20: Unused
(0)
;-----
;      Setup of port 3
;-----
      MOV     P3,     #0000000B           ;Sets P3 to its initial value.
;|||++++----- P34/P33/P32/P31/P30: Unused (0)
;++++----- <Fixed to 000>
      MOV     PM3,    #1110000B           ;Sets P3 to input or output.
;|||++++----- PM34/PM33/PM32/PM31/PM30: Unused (0)
;++++----- <Fixed to 111>
;-----
;      Setup of port 4
;-----
      MOV     P4,     #0000000B           ;Sets P4 to its initial value.
;+++++----- P47/P46/P45/P44/P43/P42/P41/P40: Unused (0)
      MOV     PM4,    #0000000B           ;Sets P4 to input or output.
;+++++----- PM47/PM46/PM45/PM44/PM43/PM42/PM41/PM40:
Unused (0)
;-----
;      Setup of port 8
;-----
      MOV     P8,     #0000000B           ;Sets P8 to its initial value.
;|||++++----- P83/P82/P81/P80: Unused (0)
;++++----- <Fixed to 0000>
      MOV     PM8,    #1111000B           ;Sets P8 to input or output.
;|||++++----- PM83/PM82/PM81/PM80: Unused (0)
;++++----- <Fixed to 1111>
;-----
;      Setup of port 9
;-----
      MOV     P9,     #0000000B           ;Sets P9 to its initial value.
;|||++++----- P93/P92/P91/P90: Unused (0)
;++++----- <Fixed to 0000>
      MOV     PM9,    #1111000B           ;Sets P9 to input or output.
;|||++++----- PM93/PM92/PM91/PM90: Unused (0)
;++++----- <Fixed to 1111>
;-----
;      Setup of port 10
;-----

```

APPENDIX A PROGRAM LIST

```

MOV     P10,    #0000000B           ;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
;-----

```

**APPENDIX A PROGRAM LIST**

---

```

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, #00010000B          ;Clock operating mode
        ;|||+----- <Fixed to 0000>
        ;||+----- OSCSELS: Sets up the subsystem clock pin.
        ;|||          Operating mode of the subsystem clock pin: XT1
oscillation mode
        ;|||          P123/XT1,P124/XT2: Connected to a crystal or
ceramic resonator
        ;||+----- <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

;-----
;      Setup of the real-time counter
;-----
;      The date and time are counted and the UART transmission interval is measured
;      to transmit the values of the real-time counter via UART every second.
;-----

MOV     RTCCL, #0000000B      ;Real-time counter clock selection register
;|||||+----- RTCCL1/RTCCL0: Selects the input clock (fRTC).
;|||||      fRTC = fSUB (32.768 kHz)
;+++++----- <Fixed to 000000>

MOV     RTCC0, #00001010B     ;Real-time counter control register 0
;|||||+++----- CT2/CT1/CT0: Generates an interrupt (INTRTC) at fixed
periods of 1 second.
;|||+----- AMPM: 24-hour time
;||+----- RCLOE0: Disables the output of the RTCCL pin (32.768
kHz).
;|+----- RCLOE1: Disables the output of the RTC1HZ pin (1 Hz).
;|+----- <Fixed to 0>
;+----- RTCE: Stops the real-time counter.

MOV     RTCC1, #00001000B     ;Real-time counter control register 1
;|||||+----- RWAIT: Specifies counting.
;|||||+----- RWST: Read only
;|||||+----- <Fixed to 0>
;|||+----- RIFG: Clears the interrupt generated at fixed periods.
;||+----- WAFG: Invalid
;|+----- <Fixed to 0>
;|+----- WALIE: Does not generate an interrupt (INTRTC) when the
alarm matches.
;+----- WALE: Disables matching.

MOV     RTCC2, #00000000B     ;Real-time counter control register 2
;+|||+++----- RINTE/CT2/CT1/CT0: Does not generate an interval
interrupt.
;||+----- <Fixed to 00>
;|+----- RCKDIV: Invalid
;+----- RCLOE2: Disables the output of the RTCDIV pin.

CLR1    RTCIF    ;Clears the fixed-period interrupt request.
SET1    RTCE     ;Starts the real-time counter.

;-----;

```



```

; Wait until the subsystem clock oscillation stabilizes ;
;-----;
;An interrupt is generated at fixed periods of one second to wait until the subsystem
clock oscillation stabilizes.
;The real-time counter restarts after the first fixed-period interrupt request is issued.
HRST400:
BF      RTCIF,$HRST400      ;Waits until the first fixed-period interrupt occurs.
CLR1    RTCE                ;Stops the real-time counter.

;Initial count register settings: '08/06/01 (Sunday) 00:00:00
MOV     SEC,    #00H        ;Second: 00 seconds
MOV     MIN,    #00H        ;Minute: 00 minutes
MOV     HOUR,   #00H        ;Hour: 00 hours
MOV     WEEK,   #00H        ;Day of the week: Sunday
MOV     DAY,    #01H        ;Day: 1st
MOV     MONTH,  #06H        ;Month: June
MOV     YEAR,   #08H        ;Year: '08

CLR1    RTCIF    ;Clears the fixed-period interrupt request flag.
SET1    RTCE     ;Starts the real-time counter.

;-----;
; Setup of the low-voltage detector
;-----;
MOV     LVIS,   #00001111B   ;Low-voltage detection level selection register
;|||||++++----- LVIS3/LVIS2/LVIS1/LVIS0: The detection level is LVII15
(1.93 V ±0.1 V).
;++++----- <Fixed to 0000>

MOV     LVIM,   #10000000B   ;Low-voltage detection register
;|||||+----- LVIF: Read only
;|||||+----- LVIMD: Generates an internal interrupt signal if the
supply voltage (VDD) falls below the detection voltage (VLVI) when the voltage decreases
;||||| or if VDD becomes at least VLVI when the voltage
increases.
;|||||+----- LVISEL: Detects the supply voltage (VDD) level.
;|++++----- <Fixed to 0000>
;+----- LVION: Enables low-voltage detection.

;The system waits from setting (1) LVION to checking the voltage by using LVIF (10 µs (max.)).
;10 µs == 80CLK (8 MHz)
MOV     B,      #13          ;4CLK
HRST300:
DBNZ    B,      $HRST300    ;6CLK
;4 + 6 * 13 = 82 CLK

;-----;
; Setup of the serial interface UART6

```

```

;-----
;   The count values of the real-time counter are transmitted.
;-----

MOV    CKSR6, #0000000B    ;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: Uses TI000 input (normal operation).
        ;||||+----- 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.

CLR1   LVIIF              ;Clears low-voltage detection interrupt requests.
CLR1   LVIMK              ;Unmasks low-voltage detection interrupts.

```

```

;-----
;      Enable interrupts
;-----
      EI

;*****
;
;
;      Main processing
;
;
;*****
MAIN_LOOP:

      ;The supply voltage after a reset is released is checked by servicing a low-voltage
detection interrupt.

      ;If the supply voltage is less than or equal to the detection voltage, STOP mode is
entered.
      SETI    LVIIF                ;Enables low-voltage detection interrupt requests.

      ;*****;
      ;                                ;
      ;      RTC count value transmission processing      ;
      ;                                ;
      ;*****;
      BF      RTCIF,$LMAIN500      ;Has 1 second elapsed? → NO
      CLRI    RTCIF                ;Clears interrupt requests.
;-----
; Acquire data from the real-time counter
;-----
      DI                        ;The second to year counters stop while the values of the real-
time counter are read.

                                ;Therefore, if STOP mode is subsequently entered, the time is
not incremented.  Interrupts are disabled during reading.

LMAIN200:
      SETI    RWAIT                ;Stops the second to year counters (in counter value
read or write mode).
      BF      RWST,$LMAIN200      ;Waits until counter value read or write mode is entered.
                                ;(Waits up to one 32.768 kHz clock cycle (30.5 μs) after
RWAIT is set to 1.)

      MOV     A,SEC
      MOV     RSEC,A                ;Acquires the second data.
      MOV     A,MIN
      MOV     RMIN,A                ;Acquires the minute data.
      MOV     A,HOURL

```

**APPENDIX A PROGRAM LIST**

---

```

MOV     RHOURL,A           ;Acquires the hour data.
MOV     A,WEEK
MOV     RWEEK,A           ;Acquires the day-of-the-week data.
MOV     A,DAY
MOV     RDAY,A           ;Acquires the day data.
MOV     A,MONTH
MOV     RMOTH,A          ;Acquires the month data.
MOV     A,YEAR
MOV     RYEAR,A         ;Acquires the year data.

LMAIN300:
CLR1    RWAIT           ;Specifies that the second to year counters operate.
BT      RWST,$LMAIN300  ;Waits until counter value read or write mode is
canceled.

EI      ;Finishes reading and enables interrupts.

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

LMAIN500:
NOP     ;The acknowledgment of interrupt requests is held pending while
;a manipulation instruction is executed for registers that control
interrupts.

;An NOP instruction is executed to prevent interrupts from being
disabled.
;*****;
;           ;
;   Different types of main processing           ;
;           ;
;*****;

;Any other main processing is performed here.

BR      MAIN_LOOP

;*****;
;
;   Creation and transmission of UART6 data
;
;-----
;   [ IN ] RSEC,RMIN,RHOURL,RWEEK,RDAY,RMONTH,RYEAR: RTC count values
;   [ OUT ] None
;
;   The values read from the real-time counter are converted to
;   ASCII code, set to the transmit buffer, and then transmitted.

```

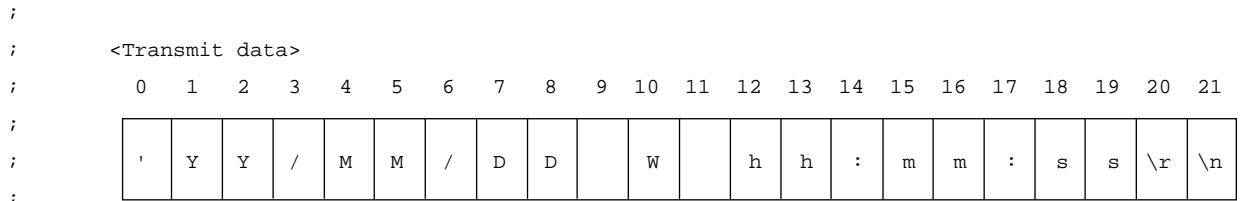
**APPENDIX A PROGRAM LIST**

---

```

;       If low voltage is detected before transmission starts, UART6 is set up again.
;       If low voltage is detected while the system waits until transmission ends, UART6
transmission is suspended.

```



(Y: Year, M: Month, D: Day, W: Day-of-the-week, h: Hour, m: Minute, s: Second)

\*\*\*\*\*

```

SUART6TX:
;-----
;   Processing to create UART6 transmit data in the transmit buffer
;-----
MOV     RTXBUF,#''''           ;[0]Saves the quotation mark (used to display the
abbreviated year).

MOV     A,RYEAR                ;Acquires the year data.
CALL    !SNUM2CHR              ;Converts the year data to ASCII code.
MOV     (RTXBUF+1),A           ;[1]Saves the higher bit of the year data.
MOV     A,X
MOV     (RTXBUF+2),A           ;[2]Saves the lower bit of the year data.

MOV     (RTXBUF+3),# '/'       ;[3]Saves the forward slash (used to delimit the date).

MOV     A,RMOTH                ;Acquires the month data.
CALL    !SNUM2CHR              ;Converts the month data to ASCII code.
MOV     (RTXBUF+4),A           ;[4]Saves the higher bit of the month data.
MOV     A,X
MOV     (RTXBUF+5),A           ;[5]Saves the lower bit of the month data.

MOV     (RTXBUF+6),# '/'       ;[6]Saves the forward slash (used to delimit the date).

MOV     A,RDAY                 ;Acquires the day data.
CALL    !SNUM2CHR              ;Converts the day data to ASCII code.
MOV     (RTXBUF+7),A           ;[7]Saves the higher bit of the day data.
MOV     A,X
MOV     (RTXBUF+8),A           ;[8]Saves the lower bit of the day data.

MOV     (RTXBUF+9),# ' '       ;[9]Saves the space.

MOV     A,RWEEK                ;Acquires the day-of-the-week data.
CALL    !SNUM2CHR              ;Converts the day-of-the-week data to ASCII code.
MOV     (RTXBUF+10),A          ;[10]Saves the day-of-the-week data.

MOV     (RTXBUF+11),# ' '      ;[11]Saves the space.

```

```

MOV     A,RHOUR                               ;Acquires the hour data.
CALL    !SNUM2CHR                             ;Converts the hour data to ASCII code.
MOV     (RTXBUF+12),A                         ;[12]Saves the higher bit of the hour data.
MOV     A,X
MOV     (RTXBUF+13),A                         ;[13]Saves the lower bit of the hour data.

MOV     (RTXBUF+14),#':'                      ;[14]Saves the colon (used to delimit the time).

MOV     A,RMIN                                ;Acquires the minute data.
CALL    !SNUM2CHR                             ;Converts the minute data to ASCII code.
MOV     (RTXBUF+15),A                         ;[15]Saves the higher bit of the minute data.
MOV     A,X
MOV     (RTXBUF+16),A                         ;[16]Saves the lower bit of the minute data.

MOV     (RTXBUF+17),#':'                      ;[17]Saves the colon (used to delimit the time).

MOV     A,RSEC                                ;Acquires the second data.
CALL    !SNUM2CHR                             ;Converts the second data to ASCII code.
MOV     (RTXBUF+18),A                         ;[18]Saves the higher bit of the second data.
MOV     A,X
MOV     (RTXBUF+19),A                         ;[19]Saves the lower bit of the second data.

MOV     (RTXBUF+20),#0DH                      ;[20]Saves the carriage return.
MOV     (RTXBUF+21),#0AH                      ;[21]Saves the line feed.

;-----
;           UART6 data transmission
;-----

BF      FUARTSTOP,$JU6TX500                  ;Has low voltage been detected? → NO

;==== Set up the serial interface UART6 again ====
CLR1    POWER6                               ;Disables the internal operation clock.
CLR1    TXE6                                 ;Disables transmission.

DI                                             ;Disables interrupts while setting up UART6 again.
SET1    TXE6                                 ;Enables transmission.
SET1    POWER6                               ;Enables the internal operation clock.
CLR1    FUARTSTOP                            ;Clears the UART transmission suspension flag.
EI                                             ;Enables interrupts.

JU6TX500:
;===== Start transmission =====
MOV     B,#22                                ;Sets up the transmission counter.
MOVW    HL,#RTXBUF

JU6TX600:
CLR1    STIF6                                ;Clears interrupt requests.
MOV     A,[HL]                               ;Acquires transmit data from the transmit buffer.

```

## APPENDIX A PROGRAM LIST

```

        MOV     TXB6,A                ;Transmits the data.
JU6TX700:
        BT      FUARTSTOP,$JU6TX800  ;Is the UART6 transmission suspension flag enabled? →
YES: Transmission ends immediately.
        BF      STIF6,$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

;*****
;
;   Convert the BCD code to ASCII code
;
;-----
;   [ IN ] A: Numeral (BCD)
;   [ OUT ] A: Higher 4 bits of the ASCII code
;           X: Lower 4 bits of the ASCII code
;*****
SNUM2CHR:
        MOV     X,A
        AND     A,#0FH                ;Acquires the lower 4 bits.
        ADD     A,#'0'                ;Converts the lower 4 bits to ASCII code.
        XCH     A,X                  ;Saves the return value and acquires the value to be converted.

        AND     A,#0F0H                ;Acquires the higher 4 bits.
        ROL     A,1                    ;Acquires the higher 4 bits by shifting them to the lower 4 bits.
        ROL     A,1
        ROL     A,1
        ROL     A,1
        ADD     A,#'0'                ;Converts the higher 4 bits to ASCII code.
        RET

;*****
;
;   Low-voltage detection completion interrupts (using INTLVI)
;
;-----
;   If VDD is at least or less than the detection voltage, an interrupt is generated.
;   The detection voltage is 1.93 V.
;   *Register bank 1 is used.
;*****
IINTLVI:
        SEL     RB1                    ;Specifies the register bank by servicing an interrupt.

```

```

HLVI100:
    BF    LVIF,$HLVI600        ;Has VDD fallen below 1.93 V? → YES

    SET1  FUARTSTOP            ;Enables the UART suspension flag.

HLVI300:
    BF    RWST,$HLVI400        ;If the real-time counter is being read
    CLR1  RWAIT                ;Cancels reading and then enters STOP mode.
    BR    HLVI300              ;Waits until reading has been canceled.

HLVI400:
    STOP                                ;Shifts to STOP mode.
    BR    HLVI100              ;Returns from STOP mode using INTVLI.

HLVI600:
    ;STOP mode is canceled because VDD is at least 1.93 V.

HLVI900:
    RETI

end

```



• main.c (C language version)

```

/*****
 NEC Electronics      78K0/Lx3 Series

*****
 78K0/LF3 Series      Sample program
*****
 Continuing to Operate Real-Time Counter in Low-Voltage Status
*****
 [History]
 2008.5.--           Newly created
*****

 [Overview]

```

This sample program continues to operate the real-time counter in the low-voltage status. The values of the count register of the real-time counter are acquired every second and transmitted via the serial interface UART6. If the supply voltage falls below 1.93 V, STOP mode is entered. If the supply voltage exceeds 1.93 V, the processing returns from STOP mode. 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 the real-time counter
- Setting up the low-voltage detector
- Setting up the serial interface UART6
- Specifying interrupt masking

<Primary main processing>

- Stopping the real-time counter
- Acquiring the values of the count register of the real-time counter
- Specifying the real-time counter to count
- Processing to create UART6 transmit data and transmitting the data

<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

<Low-voltage detection interrupts (using INTLVI)>

- Specifying the real-time counter to count
- Executing the STOP instruction

```

*****/
#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 STOP                              /* Enables the inclusion of STOP instructions.
*/
#pragma NOP                                /* Enables the inclusion of NOP instructions. */
#pragma interrupt INTLVI fn_intlvi RB1    /* Declares the interrupt function: INTLVI */

/*=====

Function prototype declarations

=====*/
static void fn_UART6_Tx(void);            /* Processing to create UART6 transmit data and
transmitting the data */

/*=====

RAM definitions

=====*/
/* Variables used to acquire data from the real-time counter */
unsigned char    ucSec                    /* Second */
                ,ucMin                    /* Minute */
                ,ucHour                   /* Hour */
                ,ucWeek                   /* Day of the week */
                ,ucYear                   /* Year */
                ,ucMonth                   /* Month */
                ,ucDay;                   /* Day */

unsigned char ucTxBuffer[22];            /* Transmit data buffer */
unsigned char ucTxBufferCounter;        /* Transmission counter */

```

```

boolean bUARTStop;                                /* UART transmission suspension flag */

/*****

Initialization processing after a reset release

*****/
void hdwinit(void)
{
    unsigned short temp;                          /* Work area */

    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. */
/*|||++++----- P34/P33/P32/P31/P30: Unused (0) */
/*+++----- <Fixed to 000> */
PM3 = 0b11100000;        /* Sets P3 to input or output. */
/*|||++++----- PM34/PM33/PM32/PM31/PM30: Unused (0) */
/*+++----- <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 */

```

**APPENDIX A PROGRAM LIST**

```

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 =0b00010000;      /* Clock operating mode */
/*|||++++----- <Fixed to 0> */
/*||+----- OSCSELS: Sets up the subsystem clock pin. */
/*|||
Operating mode of the subsystem clock pin: XT1
oscillation mode */
/*|||
P123/XT1,P124/XT2: Connected to a crystal or
ceramic resonator */
/*|+----- <Fixed to 0> */
/*++----- EXCLK/OSCSEL: */
/*
Operating mode of the high-speed system clock pin:
Input port mode */
/*
P121/X1,P122/X2/EXCLK: Input port */

```

APPENDIX A PROGRAM LIST

```

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 0> */

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 */

/*-----
Setup of the real-time counter
-----

The date and time are counted and the UART transmission interval is measured
to transmit the values of the real-time counter via UART every second.
-----*/

RTCCL = 0b00000000; /* Real-time counter clock selection register */
/* ||| || |+----- RTCCL1/RTCCL0: Selects the input clock (fRTC). */
/* ||| || fRTC = fSUB (32.768 kHz) */
/* +----- <Fixed to 000000> */

RTCC0 = 0b00001010; /* Real-time counter control register 0 */
/* ||| || |+----- CT2/CT1/CT0: Generates an interrupt (INTRTC) at fixed
periods of 1 second. */
/* || |+----- AMPM: 24-hour time */
/* || |+----- RCLOE0: Disables the output of the RTCCL pin (32.768
kHz). */
/* |+----- RCLOE1: Disables the output of the RTC1HZ pin (1 Hz).
*/
/* |+----- <Fixed to 0> */
/* +----- RTCE: Stops the real-time counter. */

```

```

RTCC1 = 0b00001000;          /* Real-time counter control register 1 */
/* ||| ||| |+----- RWAIT: Specifies counting. */
/* ||| ||| |+----- RWST: Read only */
/* ||| ||| |+----- <Fixed to 0> */
/* ||| ||| |+----- RIFG: Clears the interrupt generated at fixed periods.
*/

/* |+----- WAFG: Invalid */
/* |+----- <Fixed to 0> */
/* |+----- WALIE: Does not generate an interrupt (INTRTC) when
the alarm matches. */
/* +----- WALE: Disables matching. */

RTCC2 = 0b00000000;          /* Real-time counter control register 2 */
/* + ||| ||| +----- RINTE/CT2/CT1/CT0: Does not generate an interval
interrupt. */

/* || +----- <Fixed to 00> */
/* |+----- RCKDIV: Invalid */
/* +----- RCLOE2: Disables the output of the RTCDIV pin. */

RTCIF = 0;          /* Clears the fixed-period interrupt request. */
RTCE = 1;          /* Starts the real-time counter. */

/*-----*/
/* Wait until the subsystem clock oscillation stabilizes */
/*-----*/
/* An interrupt is generated at fixed periods of one second to wait until the subsystem
clock oscillation stabilizes. */
/* The real-time counter restarts after the first fixed-period interrupt request is
issued. */
while(!RTCIF)
{
    NOP();
}
RTCIF = 0;          /* Clears the fixed-period interrupt request. */
RTCE = 0;          /* Stops the real-time counter. */

/* Initial count register settings: '08/06/01 (Sunday) 00:00:00 */
SEC = 0;          /* Second: 00 seconds */
MIN = 0;          /* Minute: 00 minutes */
HOUR = 0;          /* Hour: 00 hours */
WEEK = 0;          /* Day of the week: Sunday */
DAY = 1;          /* Day: 1st */
MONTH = 6;          /* Month: June */
YEAR = 8;          /* Year: '08 */

RTCE = 1;          /* Starts the real-time counter. */

```

```

/*-----
Setup of the low-voltage detector
-----*/

LVIS = 0b00001111;          /* Low-voltage detection level selection register */
/*|||||++++----- LVIS3/LVIS2/LVIS1/LVIS0: The detection level is
VLVI15 (1.93 V ±0.1 V). */
/*++++----- <Fixed to 0000> */

LVIM = 0b10000000;          /* Low-voltage detection register */
/*|||||+----- LVIF: Read only */
/*|||||+----- LVIMD: Generates an internal interrupt signal if the
supply voltage (VDD) falls below the detection voltage (VLVI) when the voltage decreases */
/*|||||                          or if VDD becomes at least VLVI when the voltage
increases. */

/*|||||+----- LVISEL: Detects the supply voltage (VDD) level. */
/*|++++----- <Fixed to 0000> */
/*+----- LVION: Enables low-voltage detection. */

/* The system waits from setting (1) LVION to checking the voltage by using LVIF (10 µs (max.)).
*/
/* 10 µs == 80CLK (8 MHz) */
for(temp = 2; temp > 0; temp--)
{
    NOP();
}

/*-----
Setup of the serial interface UART6
-----

The count values of the real-time counter are transmitted.
-----*/

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 */
/*|+----- SBT6: Unused */
/*|+----- SBRT6: Read only */
/*+----- SBRF6: Unused */

ISC = 0b00001000;           /* Controls switching the input. */
/*|||||+----- ISC0: Unused */
/*|||||+----- ISC1: Uses TI000 input (normal operation). */
/*||||+----- 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. */

LVIIIF = 0;                /* Clears low-voltage detection interrupt requests. */
LVIMK = 0;                 /* Unmasks low-voltage detection interrupts. */

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

/*****

Main loop

*****/

void main(void)
{
    /* The supply voltage after a reset is released is checked by servicing a low-voltage
detection interrupt. */
    /* If the supply voltage is less than or equal to the detection voltage, STOP mode is
entered. */
    LVIIIF = 1;             /* Enables low-voltage detection interrupt requests. */

    while(1){

```

```

/*****
/*
/*      RTC count value transmission processing      */
/*
/*
/*****
if(RTCIF)
{ /* 1 second has elapsed. */
    RTCIF = 0;      /* Clears interrupt requests. */
    /*-----*/
    /* Acquire data from the real-time counter */
    /*-----*/
    DI(); /* The second to year counters stop while the values of the
real-time counter are read. */

        /* Therefore, if STOP mode is subsequently entered, the time is
not incremented. Interrupts are disabled during reading. */

        do{
            RWAIT = 1; /* Stops the second to year counters (in counter
value read or write mode). */
        }while(!RWST); /* Waits until counter value read or write mode is
entered. */

            /* (Waits up to one 32.768 kHz clock cycle (30.5 µs)
after RWAIT is set to 1.) */

            ucSec = SEC; /* Acquires the second data. */
            ucMin = MIN; /* Acquires the minute data. */
            ucHour = HOUR; /* Acquires the hour data. */
            ucWeek = WEEK; /* Acquires the day-of-the-week data. */
            ucDay = DAY; /* Acquires the day data. */
            ucMonth = MONTH; /* Acquires the month data. */
            ucYear = YEAR; /* Acquires the year data. */

            do{
                RWAIT = 0; /* Specifies that the second to year counters
operate. */

            }while(RWST); /* Waits until counter value read or write mode is
canceled. */

            EI(); /* Finishes reading and enables interrupts. */

            /*-----*/
            /* Creation and transmission of UART6 data */
            /*-----*/
            fn_UART6_Tx();
        }

    NOP(); /* The acknowledgment of interrupt requests is held pending
while */

```

```

        /* a manipulation instruction is executed for registers that
control interrupts. */

        /* An NOP instruction is executed to prevent interrupts from
being disabled. */

        /*****
        /*
        /*          Different types of main processing          */
        /*
        /*
        /*****

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

```

/\*-----\*/

Creation and transmission of UART6 data

-----

[ IN ] None  
[ OUT ] None

The values read from the real-time counter are converted to ASCII code, set to the transmit buffer, and then transmitted. If low voltage is detected before transmission starts, UART6 is set up again. If low voltage is detected while the system waits until transmission ends, UART6 transmission is suspended.

<Transmit data>

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

'	Y	Y	/	M	M	/	D	D		W		h	h	:	m	m	:	s	s	\r	\n
---	---	---	---	---	---	---	---	---	--	---	--	---	---	---	---	---	---	---	---	----	----

(Y: Year, M: Month, D: Day, W: Day-of-the-week, h: Hour, m: Minute, s: Second)

\*\*\*\*\*

```

static void fn_UART6_Tx(void)
{
    /*****
    /*
    /*          Processing to create UART6 transmit data          */
    /*
    /*
    /*****

    ucTxBuffer[0] = '\';          /* Quotation mark (used to
display the abbreviated year) */
    ucTxBuffer[1] = (( ucYear & ( 0xF0 )) >> 4 ) + '0';          /* Higher bits of the year data
(converted to ASCII code) */
    ucTxBuffer[2] = ( ucYear & ( 0x0F )) + '0';          /* Lower bits of the year data
(converted to ASCII code) */

```

```

        ucTxBuffer[3] = '/';                /* Forward slash (used to
delimit the date) */
        ucTxBuffer[4] = (( ucMonth & ( 0xF0 )) >> 4 ) + '0'; /* Higher bits of the month data
(converted to ASCII code) */
        ucTxBuffer[5] = ( ucMonth & ( 0x0F )) + '0';        /* Lower bits of the month data
(converted to ASCII code) */
        ucTxBuffer[6] = '/';                /* Forward slash (used to
delimit the date) */
        ucTxBuffer[7] = (( ucDay & ( 0xF0 )) >> 4 ) + '0'; /* Higher bits of the day data
(converted to ASCII code) */
        ucTxBuffer[8] = ( ucDay & ( 0x0F )) + '0';          /* Lower bits of the day data
(converted to ASCII code) */
        ucTxBuffer[9] = ' ';                /* Space */
        ucTxBuffer[10] = ( ucWeek & ( 0x0F )) + '0';        /* Day-of-the-week data */
        ucTxBuffer[11] = ' ';              /* Space */
        ucTxBuffer[12] = (( ucHour & ( 0xF0 )) >> 4 ) + '0'; /* Higher bits of the hour data
(converted to ASCII code) */
        ucTxBuffer[13] = ( ucHour & ( 0x0F )) + '0';        /* Lower bits of the hour data
(converted to ASCII code) */
        ucTxBuffer[14] = ':';              /* Colon (used to delimit the
time) */
        ucTxBuffer[15] = (( ucMin & ( 0xF0 )) >> 4 ) + '0'; /* Higher bits of the minute
data (converted to ASCII code) */
        ucTxBuffer[16] = ( ucMin & ( 0x0F )) + '0';        /* Lower bits of the minute data
(converted to ASCII code) */
        ucTxBuffer[17] = ':';              /* Colon (used to delimit the
time) */
        ucTxBuffer[18] = (( ucSec & ( 0xF0 )) >> 4 ) + '0'; /* Higher bits of the second
data (converted to ASCII code) */
        ucTxBuffer[19] = ( ucSec & ( 0x0F )) + '0';        /* Lower bits of the second data
(converted to ASCII code) */
        ucTxBuffer[20] = '\r';              /* Carriage return */
        ucTxBuffer[21] = '\n';              /* Line feed */

        /*****
        /*
        /*          UART6 data transmission          */
        /*
        /*
        /*****
        /*-----*/
        /* Set up the serial interface UART6 again */
        /*-----*/
        if(bUARTStop) /* If low voltage has been detected, UART6 must be set up again. */
        {
            POWER6 = 0;          /* Disables the internal operation clock. */
            TXE6 = 0;           /* Disables transmission. */

            DI();                /* Disables interrupts while setting up UART6 again. */

```

```

TXE6 = 1;          /* Enables transmission. */
POWER6 = 1;       /* Enables the internal operation clock. */
bUARTStop = 0;   /* Clears the UART transmission suspension flag. */
EI();            /* Enables interrupts. */
}

/*-----*/
/*          Start 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. */
        if(bUARTStop)
            /* If the transmission suspension flag is enabled during transmission,
transmission is suspended. */
                ucTxBufferCounter = sizeof(ucTxBuffer);
                break;
    }
}
}

}

/*****

Low-voltage detection completion interrupts (using INTLVI)

-----

If VDD is at least or less than the detection voltage, an interrupt is generated.
The detection voltage is 1.93 V.
*****/
__interrupt void fn_intlvi(void)
{
    while(LVIF == 1) /* Enters STOP mode if VDD < 1.93 V. */
    {
        bUARTStop = 1; /* Enables the UART6 transmission suspension flag. */

        while(RWST) /* If the real-time counter is being read */
        {
            RWAIT = 0; /* Cancels reading and then enters STOP mode. */
        }
        STOP(); /* Shifts to STOP mode. */
    }

    /* STOP mode is canceled if VDD is at least 1.93 V. */
}

```

## 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 Esplandiú, 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/>