To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 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 1st, 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.

RENESAS

Application Note

78K0R/LH3

Sample Program (Touch Sensor)

Touch Sensor Program

This document generally describes the operation and use of a sample program. This sample program is used for detecting where a touch sensor has been touched by using a microcontroller pin and internal timer to measure the capacitance of the touch sensor's capacitors. The program then displays the touch location on an LCD.

Target device

78K0R/LH3 microcontroller

CONTENTS

Document No. U20227EJ1V0AN00 (1st edition)
Date Published March 2010 N

- The information in this document is current as of February, 2010. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, 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, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- 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.

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

(M8E0909E)

CHAPTER 1 OVERVIEW

This sample program is used to implement a touch sensor feature by using a microcontroller pin and timer to measure the capacitance of the touch sensor's capacitors. The touch sensor's capacitors are charged and the time it takes to charge them is handled as the capacitance. A touch is detected by using the phenomenon whereby the touch sensor capacitance increases when the touch sensor is touched by a fingertip.

The location of the touch on the matrix is calculated by using the information from two pins and that location is displayed on the LCD.

CHAPTER 2 LOCATION DETECTION ALGORITHM

This chapter describes the location detection algorithm used by the sample program. The touch location is detected by using a procedure whereby each of the touch sensor's capacitors is charged and the time it takes to charge them is measured. Changes in the capacitance indicate that a touch has occurred.

2.1 Measuring Capacitance

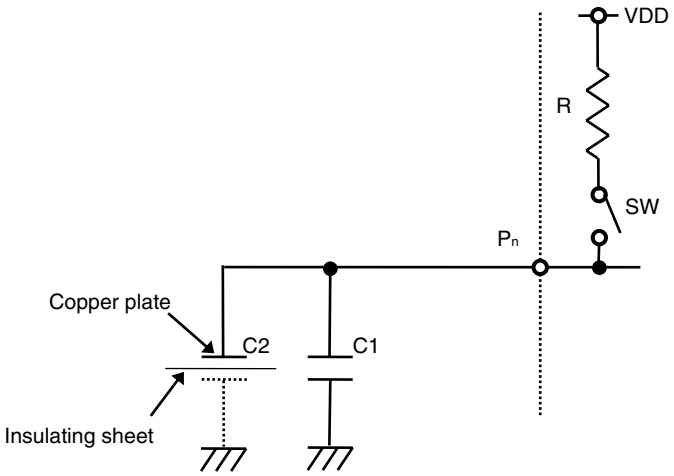


Figure 2-1-1. Example of Touch Point Circuit

Figure 2-1-1 shows the touch point circuit.

Capacitor C1 is a floating capacitor that holds the charge of parts of the circuit such as the wiring leading to the pad. Capacitor C2 is a capacitor that holds the charge generated when the insulating sheet on top of the copper plate is touched by a finger. When this sheet has not been touched, the capacitance of C2 is almost 0. When this sheet is touched, capacitor C2 is charged and the combined capacitance of C1 and C2 increases.

P_n is a microcontroller I/O pin. R is a pull-up resistor on the microcontroller. SW is a switch on the microcontroller that is on when the pin is specified as an input pin and off when the pin is specified as an output pin.

The capacitance is measured as follows: A low-level signal is output to P_n and C1 and C2 fully discharge. P_n is then specified as an input pin and SW turns on. C1 and C2 begin to charge. If a touch has occurred, C2 takes a long time to charge. If a touch has not occurred, C2 does not take a long time to charge. The occurrence of a touch is therefore judged by how long it takes C2 to charge.

(1) Measuring the charging time of the capacitors

The circuit shown in Figure 2-1-1 is used to measure the charging time of the capacitors. The charging time is measured by measuring how long the P_n signal is low level, which is performed by measuring the input pulse interval of channel 6 of timer array unit 0 and detecting the rising edge of $INTP_n$. For the detailed settings of channel 6 of timer array unit 0, see **5.1 Initialization of Peripherals to Be Used**.

Remark The capacitance of the capacitors (the area of the copper plate) is a value that ensures that channel 6 of timer array unit 0 does not overflow when the charging time is measured and that makes it possible to judge whether C2 has been charged or not.

The procedure for measuring the charging time is outlined below.

<1> P_n is set to low-level output and the capacitors discharge.
<2> P_n is specified to be used as the $INTP_n$ pin (with $INTP_n$ set to “detect the rising edge”) and is specified as an input pin in order to enable its pull-up resistor.
<3> When P_n is specified as an input pin, its pull-up resistor is enabled and the capacitors begin to charge via the pull-up resistor.
<4> When the capacitors being charging in <3> above, the operation of channel 6 of timer array unit 0 is enabled and the timer begins measuring the charging time.
<5> The system waits until the capacitors are charged (until the $INTP_n$ interrupt occurs).
<6> The measured charging time is obtained from channel 6 of timer array unit 0.
<7> The operation of channel 6 of timer array unit 0 is disabled.

(2) Calculating the touched location

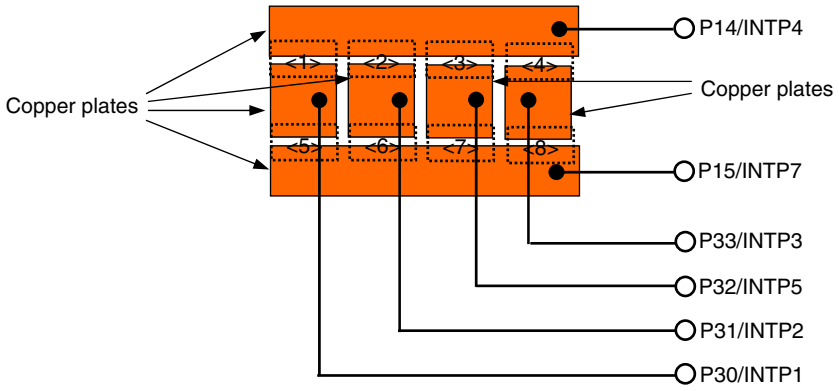


Figure 2-1-2. Example Touch Point Matrix

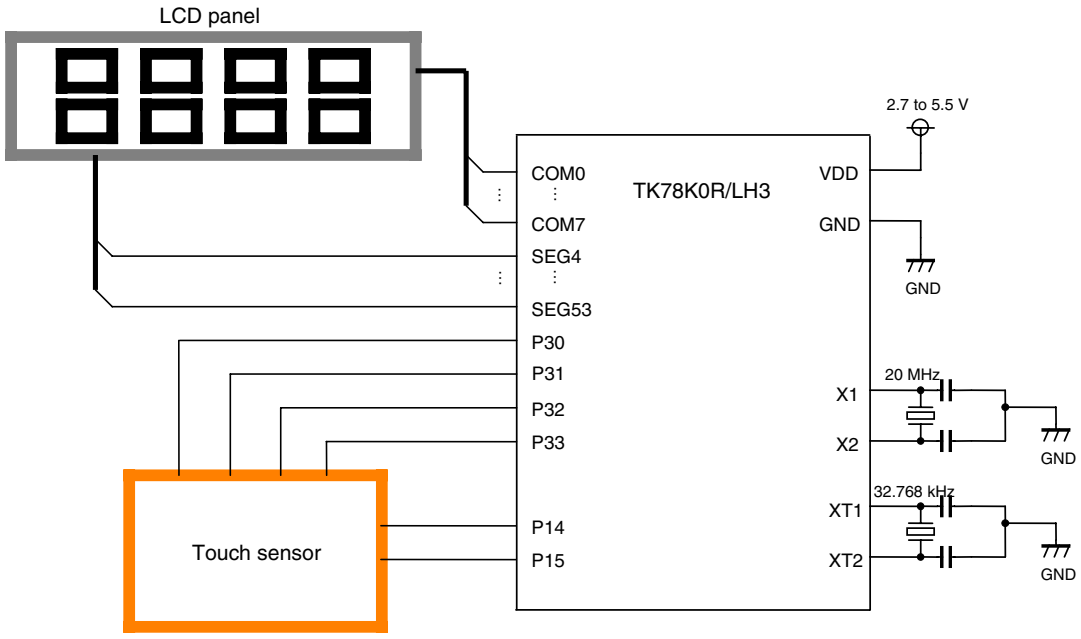
A copper plate is allocated to each pin as shown in the example touch point matrix in Figure 2-1-2 above.
The capacitance of these copper plates is measured according to the time it takes to charge the capacitors of each pin. If this charging time is longer than the reference value, a touch is judged to have occurred.
A reference value is specified as a variable for each pin. Each time the capacitance is measured, the measured value is compared with the reference value. If the measured value is shorter than the reference value, the reference value is overwritten by the measured value. In this way, the reference value is always the shortest measured value.
Note that any measured value that is shorter than the reference value will replace the reference value, even if the measured value is the product of noise or some other anomaly. In this case, it will not be possible to recover a correct reference value. It is therefore important to take measures to prevent noise and other circuit anomalies.
The difference between the reference value and the charging time, which is used to determine whether a touch has occurred or not, differs depending on the area of the copper plate (capacitance). In this sample program, a touch is judged to have occurred if the charging time is more than 0.25 μs longer than the reference value.
After the charging time of all the pins has been measured, the touch location is calculated by combining the information of the pins at which a touch was judged to have occurred.
For example, if a touch is judged to have occurred at pins P14 and P30, the location of the touch is <1> on the example touch point matrix in Figure 2-1-2.
In this sample program, a touch is judged to have occurred on the matrix when a touch has occurred at two pins, and the touch is determined as being at locations <1> to <8> in Figure 2-1-2 according to the various 2-pin combinations.
If a touch is judged to have occurred at two or more pins, it is deemed that two locations have been touched at the same time. Moreover, if the number of pins at which a touch is judged to have occurred is 0 or 1, it is judged that no touch has occurred on the matrix.
Touch location detection is executed every 15 ms. New touch location information is compared with the previous information. If a match occurs, that information becomes the final location information.

CHAPTER 3 CIRCUIT DIAGRAM

This chapter provides a circuit diagram and describes the devices used in this sample program other than the microcontroller.

3.1 Circuit Diagram

A circuit diagram is shown below.



Cautions 1. Use the microcontroller at a voltage in the range of 2.7 V ≤ V_{DD} ≤ 5.5 V.

2. Make EV_{DD}, AV_{DD0}, and AV_{DD1} have the same potential as V_{DD}.

3. Make AV_{SS} have the same potential as EV_{SS} and V_{SS}, and connect it directly to GND.

4. For details of these settings, see the TK-78K0R/LH3+LCD User's Manual.

3.2 Used Device Other than Microcontroller

The following device is used in addition to the microcontroller:

(1) LCD

Use the LCD provided with TK-78K0R/LH3+LCD.

CHAPTER 4 SOFTWARE

This chapter describes the files included in the compressed file to be downloaded, internal peripheral functions of the microcontroller to be used, and initial settings and provides an operation overview of the sample program.

4.1 Included Files

The following table shows the files included in the compressed file to be downloaded.

File Name	Description	Compressed (*.zip) File Included	
			
main.c	C language source file for hardware initialization processing and main processing of microcontroller	●	●
displayl.c	C language source file for LCD driver initialization and LCD display processing		
op.asm	Assembler source file for setting the option byte (This file is used for setting up the watchdog timer and internal low-speed oscillator.)	●	●
78K0RLx3_sample_program.prw	Work space file for integrated development environment PM plus		●
78K0RLx3_sample_program.prj	Project file for integrated development environment PM plus		●

Remark  : Only the source file is included.

 : The files to be used with integrated development environment PM plus are included.

4.2 Internal Peripheral Functions to Be Used

The following describes peripheral functions that are built into the microcontroller that is used by the sample program.

- High-speed system clock oscillator
  For CPU clock and peripheral hardware clock
- Channel 4 of timer array unit 0 (TAU0)
  This channel is used as an interval timer to make the system wait for the operation of the hardware to stabilize and for the floating capacitors to charge.
- Channel 6 of timer array unit 0 (TAU0)
  This channel is used in capture mode to measure the rise time of P14 and P15, and P30 to P33.
- P14 and P15, and P30 to P33
  These pins are connected to the touch sensor.
- INTP1 to INTP5 and INTP7
  These pins are used to input a charge from the touch sensor and detect a rising edge.
- LCD controller/driver
  This displays measured count values on the LCD on the board.

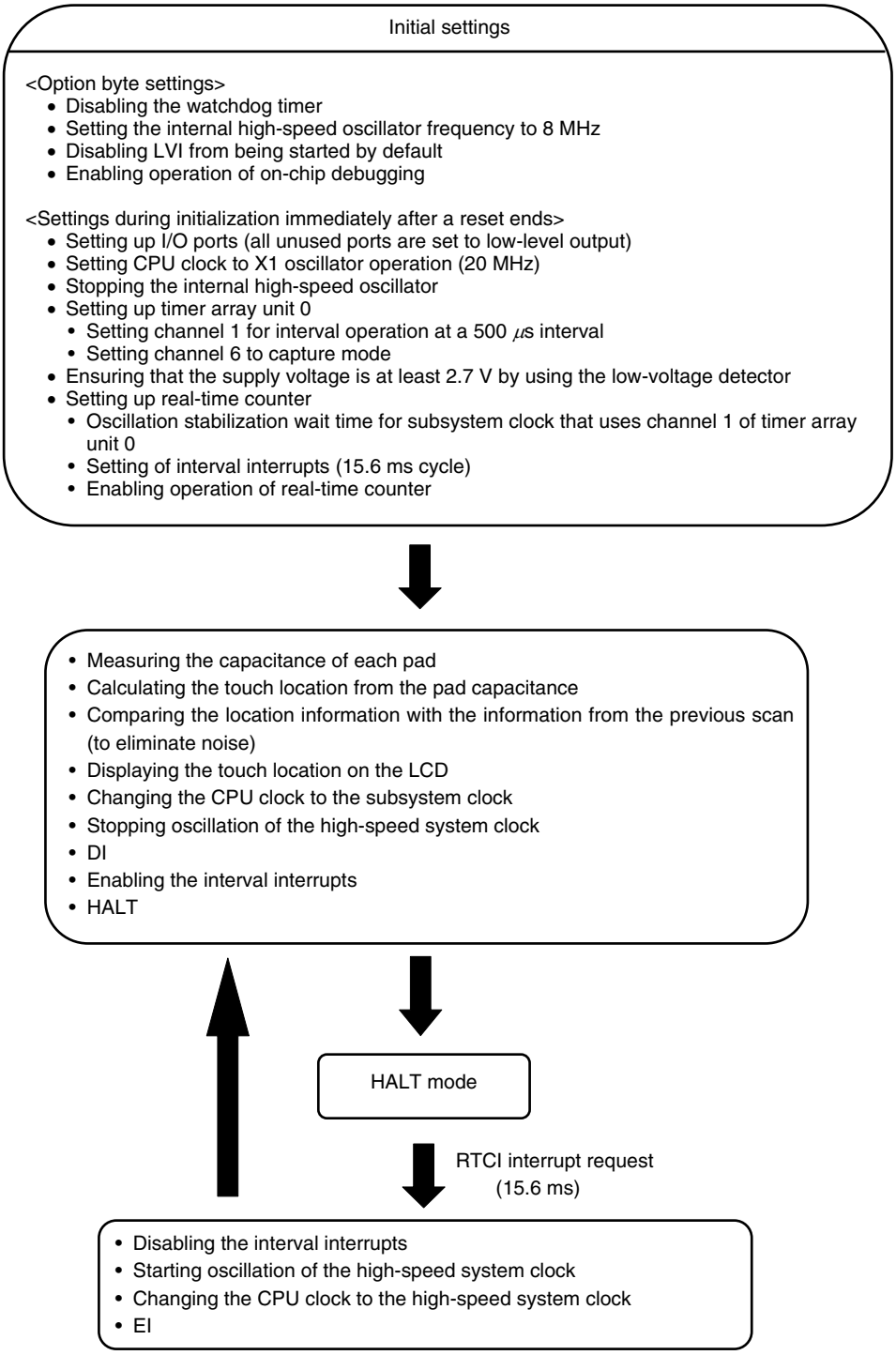
4.3 Initial Settings and Operation Overview

In this sample program, initialization sets the port settings, clock frequency selection, and the settings for timer array unit 0, the LCD controller/driver, etc. Touch sensor detection processing starts after the initial settings have been specified for the peripherals.

First, the floating capacitor is charged by switching the target pin to output mode, outputting a low-level signal, and waiting for a given period of time. The pin is then switched to input mode and the floating capacitor discharges. At the same time, channel 6 of timer array unit 0 starts counting. When a rising edge is detected at the target pin, channel 6 of timer array unit 0 stops counting and its count value is obtained. This processing is repeated for P3.0 to P3.3, and P1.4 and P1.5.

The touch location is calculated from the results of touch detection at P3.0 to P3.3, and P1.4 and P1.5. This location is then displayed on the LCD.

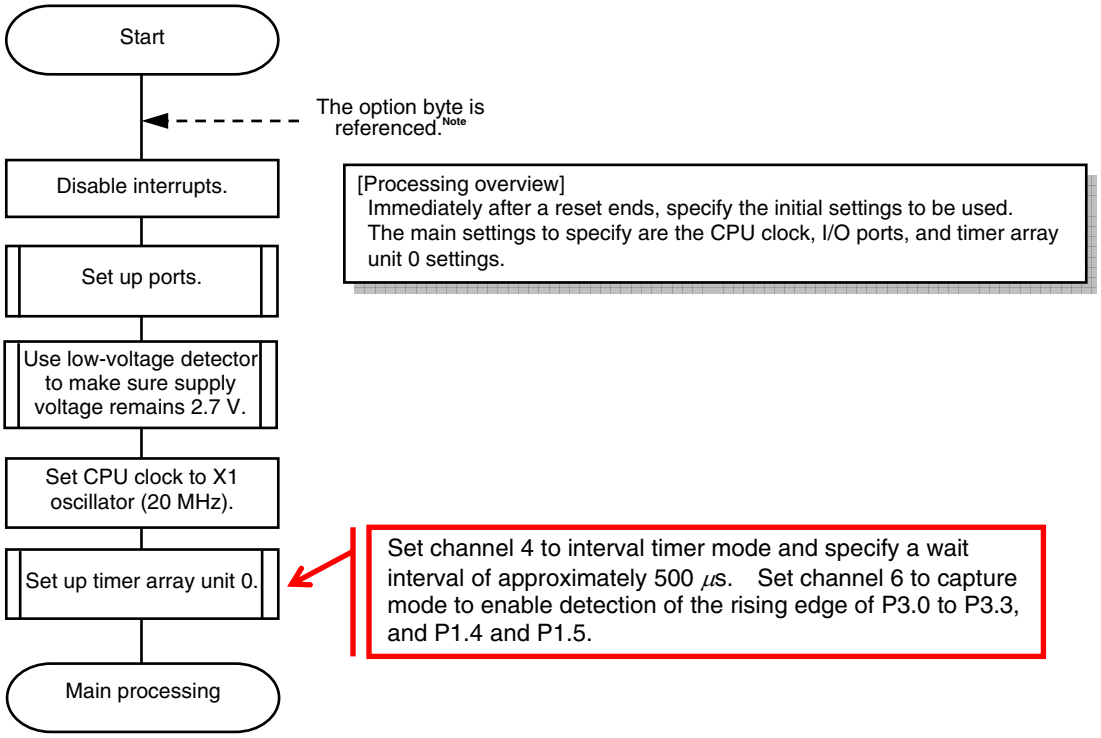
The details are described in the state transition diagram shown below.


Initial settings
<Option byte settings>
• Disabling the watchdog timer
• Setting the internal high-speed oscillator frequency to 8 MHz
• Disabling LVI from being started by default
• Enabling operation of on-chip debugging
<Settings during initialization immediately after a reset ends>
• Setting up I/O ports (all unused ports are set to low-level output)
• Setting CPU clock to X1 oscillator operation (20 MHz)
• Stopping the internal high-speed oscillator
• Setting up timer array unit 0
• Setting channel 1 for interval operation at a 500 μs interval
• Setting channel 6 to capture mode
• Ensuring that the supply voltage is at least 2.7 V by using the low-voltage detector
• Setting up real-time counter
• Oscillation stabilization wait time for subsystem clock that uses channel 1 of timer array unit 0
• Setting of interval interrupts (15.6 ms cycle)
• Enabling operation of real-time counter
• Measuring the capacitance of each pad
• Calculating the touch location from the pad capacitance
• Comparing the location information with the information from the previous scan (to eliminate noise)
• Displaying the touch location on the LCD
• Changing the CPU clock to the subsystem clock
• Stopping oscillation of the high-speed system clock
• DI
• Enabling the interval interrupts
• HALT
HALT mode
RTCI interrupt request
(15.6 ms)
• Disabling the interval interrupts
• Starting oscillation of the high-speed system clock
• Changing the CPU clock to the high-speed system clock
• EI

4.4 Flow Charts

The flow charts for the sample program are shown below.

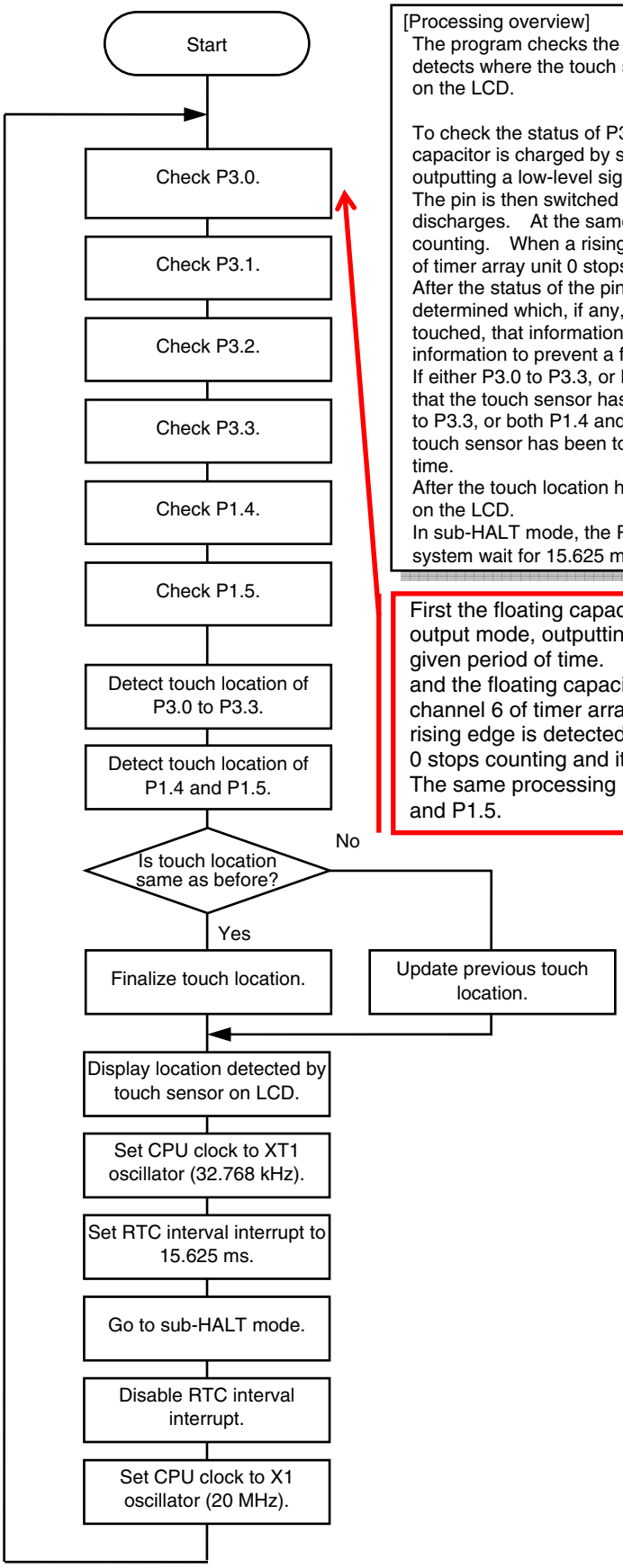
<Settings during initialization immediately after a reset ends>



Note The option byte is automatically referenced by the microcontroller immediately after a reset ends. In this sample program, the following settings are specified using the option byte:

- Disabling the watchdog timer
- Setting the internal high-speed oscillator frequency to 8 MHz
- Disabling LVI from being started by default
- Enabling operation of on-chip debugging

<Main processing>



[Processing overview]
The program checks the status of P3.0 to P3.3, and P1.4 and P1.5, detects where the touch sensor was touched, and displays that location on the LCD.

To check the status of P3.0 to P3.3, and P1.4 and P1.5, first the floating capacitor is charged by switching the target pin to output mode, outputting a low-level signal, and waiting for a given period of time. The pin is then switched to input mode and the floating capacitor discharges. At the same time, channel 6 of timer array unit 0 starts counting. When a rising edge is detected at the target pin, channel 6 of timer array unit 0 stops counting and its count value is obtained.
After the status of the pins has been checked and it has been determined which, if any, of P3.0 to P3.3, and P1.4 and P1.5 were touched, that information is compared with the previous touch detection information to prevent a false result due to chattering.
If either P3.0 to P3.3, or P1.4 or P1.5 do not record a touch, it is judged that the touch sensor has not been touched. If more than one of P3.0 to P3.3, or both P1.4 and P1.5 record touches, it is judged that the touch sensor has been touched in more than one place at the same time.
After the touch location has been calculated, that location is displayed on the LCD.
In sub-HALT mode, the RTC interval interrupt is used to make the system wait for 15.625 ms.

First the floating capacitor is charged by switching P3.0 to output mode, outputting a low-level signal, and waiting for a given period of time. P3.0 is then switched to input mode and the floating capacitor discharges. At the same time, channel 6 of timer array unit 0 starts counting. When a rising edge is detected at P3.0, channel 6 of timer array unit 0 stops counting and its count value is obtained.
The same processing is repeated for P3.1, P3.2, P3.3, P1.4, and P1.5.

CHAPTER 5 SETTING METHODS

This chapter describes initialization of peripherals to be used, and processing by the sample program.

For details of option bytes, clock frequencies, and register setting methods, see the user's manual for this product (78K0R/Lx3) and the sample program.

5.1 Initialization of Peripherals to Be Used

(1) Real-time counter initialization

The following operations are performed during real-time counter initialization.

<1> An input clock is supplied to the real-time counter.

<2> The real-time counter is set as follows:

- Real-time counter operation: Counter operation is stopped
- RTC1HZ pin output: RTC1HZ pin output (1 Hz) is prohibited
- RTCCL pin output: RTCCL pin output (32.768 kHz) is prohibited
- 12-/24-hour system: 24-hour system
- Periodic interrupt (INTRTC): Once per 0.5 second (simultaneous with second count-up)

<3> Real-time counter interrupt is disabled.

```
void    fn_InitRtc(void)
{
<1>-----
        RTCEN = 1;              /* supplies operational real-time counter (RTC) input clock.
*/

<2>-----
        RTCC0 = 0b00001010;             /* Real-Time Counter Control Register 0 */
               /*|||||+++--- : Constant-period interrupt (INTRTC) selection */
               /*|||||  : 0 0 0 : Does not use constant-period interrupt function. */
               /*|||||  : 0 0 1 : Once per 0.5 s */
               /*|||||  : 0 1 0 : Once per 1 s */
               /*|||||  : 0 1 1 : Once per 1 m */
               /*|||||  : 1 0 0 : Once per 1 hour */
               /*|||||  : 1 0 1 : Once per 1 day */
               /*|||||  : 1 1 x : Once per 1 month */
               /*|||||   x = don't care */
               /*||||| */
               /*||||+------ : Selection of 12-/24-hour system */
               /*||||  : 0 : 12-hour system */
               /*||||  : 1 : 24-hour system */
               /*|||| */
               /*|||+------- : RTCCL pin output control */
               /*|||  : 0 : Disables output of RTCCL pin (32 kHz). */
               /*|||  : 1 : Enables output of RTCCL pin (32 kHz). */
               /*||| */
               /*||+-------- : RTC1HZ pin output control */
               /*||  : 0 : Disables output of RTC1HZ pin (1 Hz). */
               /*||  : 1 : Enables output of RTC1HZ pin (1 Hz). */
               /*|| */
               /*|+--------- : Be sure to set 0 */
               /*| */
               /*+---------- : Real-time counter operation control */
               /*  : 0 : Stops counter operation. */
               /*  : 1 : Starts counter operation. */

<3>-----
        RTCMK = 1;                      /* disable RTC interrupt */
        RTCIF = 0;                      /* clear RTC interrupt request flag */
-----
}
```

(2) LCD controller/driver initialization

The following operations are performed during LCD controller/driver initialization.

<1> The LCD drive voltage generator is set to internal voltage boosting method.

<2> The SEG8 to SEG26 pins are set to segment output enable.

<3> The P50 to P57, P90 to P97, P100 to P102, and P140 to P147 pins are set to segment output.

<4> The input switch control register is set as follows:

- Schmitt trigger buffer control of TI04/SEG50/P53: Input disabled
- Schmitt trigger buffer control of TI02/SEG51/P52: Input disabled
- Schmitt trigger buffer control of RxD3/SEG53/P50: Input disabled

<5> The LCD RAM area is cleared.

<6> LCD clock control register 0 is set as follows:

- LCD source clock (f_{LCD}): f_{SUB}
- LCD clock (LCDCL) selection: $f_{LCD}/2^7$

<7> Voltage V_{LC0} is set to 5.0 V.

<8> A 2 ms wait is inserted.

<9> The LCD display mode register is set as follows:

- LCD display enable/disable: Ground level output to segment pin and common pin
- Enable/stop operation of booster circuit and capacitance splitting circuit: Operation enabled
- Display data area: Data in A pattern area is displayed
- Display mode of LCD controller/driver: Eight time divisions and 1/4 bias

<10> A 500 ms wait is inserted.

<11> LCD display is set to ON.

```
void    fn_DisplayInit(void)
{
        /*----------------------------------------*/
        /* Initialization of LCD controler/driver */
        /*----------------------------------------*/
<1>-----        LCDMD = 0b00010000;             /* LCD Mode Register */
                /*++||++++--- : Be sure to set 000000 */
                /*  || */
                /*  ++------- : LCD drive voltage generator selection */
                /*   : 0 0 : External resistance division method */
                /*   : 0 1 : Internal voltage boosting method */
                /*   : 1 0 : Capacitance split method */
                /*   : 1 1 : Setting prohibited */
<2>-----        SEGEN = 0b00011111;             /* Segment Enable Register */
                /*|||||||+--- : Control segment signal output from pins SEG8-SEG11 */
                /*||||||+---- : Control segment signal output from pins SEG12-SEG15 */
                /*|||||+----- : Control segment signal output from pins SEG16-SEG19 */
                /*||||+------ : Control segment signal output from pins SEG20-SEG23 */
                /*|||+------- : Control segment signal output from pins SEG24-SEG26 */
                /*||| : 0 : segment signal output disable */
                /*||| : 1 : segment signal output enable */
                /*||| */
                /*+++-------- : Be sure to set 000 */
<3>-----        PFALL = 0b01111111;             /* Port Function Register ALL */
                /*|||||||+--- : Pins P50-P53 port/segment output specification */
                /*||||||+---- : Pins P54-P57 port/segment output specification */
                /*|||||+----- : Pins P90-P93 port/segment output specification */
                /*||||+------ : Pins P94-P97 port/segment output specification */
                /*|||+------- : Pins P100-P102 port/segment output specification */
                /*||+-------- : Pins P140-P143 port/segment output specification */
                /*|+--------- : Pins P144-P147 port/segment output specification */
                /*| : 0 : Used the pins as port (other than segment output) */
                /*| : 1 : Used the pins as segment output */
                /*| */
                /*+---------- : Be sure to set 0 */
<4>-----        ISC =   0b00000000;             /* Input Switch Control Register */
                /*|||||||+--- : Switching external interrupt (INTP0) input */
                /*||||||| : 0 : Uses the input signal of the INTP0 pin as an external interrupt
(normal operation). */
                /*||||||| : 1 : Uses the input signal of the RXD3 pin as an external interrupt
*/
                /*|||||||      (to measure the pulse widths of the sync break field and sync
field). */
                /*||||||| */
                /*||||||+---- : ISC1 Switching channel 7 input of timer array unit TAUS */
                /*|||||| : 0 : Uses the input signal of the TI07 pin as a timer input (normal
operation). */
                /*|||||| : 1 : Input signal of RXD3 pin is used as timer input (wakeup signal
detection). */
                /*|||||| */
                /*|||||+----- : RxD3/SEG53/P50 pin schmitt-triggered buffer control */
                /*||||| : 0 : Disables input */
                /*||||| : 1 : Enables input */
                /*||||| */
                /*||||+------ : TI02/SEG51/P52 pin schmitt-triggered buffer control */
                /*|||| : 0 : Disables input */
                /*|||| : 1 : Enables input */
                /*|||| */
                /*|||+------- : TI04/SEG50/P53 pin schmitt-triggered buffer control */
                /*||| : 0 : Disables input */
                /*||| : 1 : Enables input */
                /*||| */
                /*+++-------- : Be sure to set 000 */

<5>-----        fn_DisplayAllClear();           /* clear all LCD String area */
```

```
<6>-----        LCDC0 = 0b00000011;             /* LCD Clock Control Register */
                /*|||||+++--- : LCD clock (LCDCL) selection */
                /*|||||  : 0 0 0 : fLCD/2^4 */
                /*|||||  : 0 0 1 : fLCD/2^5 */
                /*|||||  : 0 1 0 : fLCD/2^6 */
                /*|||||  : 0 1 1 : fLCD/2^7 */
                /*|||||  : 1 0 0 : fLCD/2^8 */
                /*|||||  : 1 0 1 : fLCD/2^9 */
                /*|||||  : Other than above : Setting prohibited */
                /*|||||  */
                /*++||+------ : Be sure to set 000 */
                /*  ||  */
                /*  ++------- : LCD source clock (fLCD) selection */
                /* : 0 0 : fSUB */
                /* : 0 1 : fCLK/2^6 */
                /* : 1 0 : fCLK/2^7 */
                /* : 1 1 : fCLK/2^8 */

<7>-----        VLCD = 0x0A;                    /* set LCD boost level to 5V */
                /* wait for the reference voltage setup time (2ms(min.)) */
<8>-----        fn_Wait500usBase(2000/500);

<9>-----        LCDM =  0b00100111;             /* LCD Display Mode Register */
                /*|||||+++--- : LCD controller/driver display mode selection */
                /*|||||  *When the external resistance division method is used */
                /*|||||  : 0 0 0 : Four-time-slice mode & 1/3 bias method */
                /*|||||  : 0 0 1 : Three-time-slice mode & 1/3 bias method */
                /*|||||  : 0 1 0 : Two-time-slice mode & 1/2 bias method */
                /*|||||  : 0 1 1 : Three-time-slice mode & 1/2 bias method */
                /*|||||  : 1 0 0 : Static */
                /*|||||  : 1 1 1 : Eight-time-slice mode & 1/4 bias method */
                /*|||||  */
                /*|||||  *When the internal voltage boosting method is used */
                /*|||||  : 0 0 0 : Four-time-slice mode & 1/3 bias method */
                /*|||||  : 0 0 1 : Three-time-slice mode & 1/3 bias method */
                /*|||||  : 0 1 0 : Four-time-slice mode & 1/3 bias method */
                /*|||||  : 0 1 1 : Four-time-slice mode & 1/3 bias method */
                /*|||||  : 1 0 0 : Setting prohibited */
                /*|||||  : 1 1 1 : Eight-time-slice mode & 1/4 bias method */
                /*|||||  */
                /*|||||  *When the capacitor split method is used */
                /*|||||  : 0 0 0 : Four-time-slice mode & 1/3 bias method */
                /*|||||  : 0 0 1 : Three-time-slice mode & 1/3 bias method */
                /*|||||  : 0 1 0 : Four-time-slice mode & 1/3 bias method */
                /*|||||  : 0 1 1 : Four-time-slice mode & 1/3 bias method */
                /*|||||  : 1 0 0 : Setting prohibited */
                /*|||||  : 1 1 1 : Four-time-slice mode & 1/3 bias method */
                /*|||||  */
                /*|||||  : Other than above : Setting prohibited
                /*|||||  */
                /*|||++------ : LCD display data area control */
                /*|||  : 0 0 : Display the data of an A pattern area */
                /*|||         (lower 4 bits of LCD display data memory) */
                /*|||  : 0 1 : Display the data of an A pattern area */
                /*|||         (higher 4 bits of LCD display data memory) */
                /*|||  : 1 0 : Display the data of an A pattern area and the B pattern area in
turn. */
                /*|||         (The on and off light indication which synchronized */
                /*|||          in a constant-period interrupt timing of RTC) */
                /*|||  : 1 1 : Display the data of an A pattern area and the B pattern area in
turn. */
                /*|||         (The on and off light indication which synchronized */
                /*|||          in a constant-period interrupt timing of RTC) */
                /*||| */
```

```
                /*||+-------- : Voltage boost circuit and capacitor split circuit operation
enable/disable */
                /*|| : 0 : Stops voltage boost circuit and capacitor split circuit operation
*/
                /*|| : 1 : Enables voltage boost circuit and capacitor split circuit operation
*/
                /*|| */
                /*++--------- : LCD display enable/disable */
                /* : 0 0 : Output ground level to segment/common pin */
                /* : 0 1 : Display off (all segment outputs are deselected.) */
                /* : 1 0 : Output ground level to segment/common pin */
                /* : 1 1 : Display on */

        /* software to wait for the operation stabilization time (over 500ms) */
<10>    fn_Wait500usBase(500000/500);

        SCOC = 1;                       /* output deselect level to SEG and LCD waveform to COM
<11> */
        LCDON = 1;                      /* display on */
}
```

5.2 Main Processing

The following operations are performed during main processing.

<1> Specify detection of the rising edge of P1.4 and P1.5, and P3.0 to P3.3.

<2> Initialize LCD display.

<3> Execute the following processing for P3.0. The same processing is repeated for P3.1, P3.2, P3.3, P1.4, and P1.5.

(a) Set P3.0 to low-level output and charge the floating capacitor.

(b) Specify P3.0 as an input pin and measure its rise time.

(c) If the count when the touch sensor is touched is lower than when the touch sensor is not touched, update the count for when the touch sensor is not touched.

<4> Detect touch location of P3.0 to P3.3.

<5> Detect touch location of P1.4 and P1.5.

<6> If the new results match the previous results, the detected touch sensor location is updated.

<7> If either P3.0 to P3.3, or P1.4 or P1.5 do not record a touch, it is judged that the touch sensor has not been touched.

<8> If more than one of P3.0 to P3.3, or both P1.4 and P1.5 record touches, it is judged that the touch sensor has been touched in more than one place at the same time.

<9> Display location detected by touch sensor on LCD.

<10> Change the CPU clock to the subsystem clock.

<11> Shift to sub-HALT mode and wait for approximately 15 ms.

<12> Change the CPU clock to the main clock.

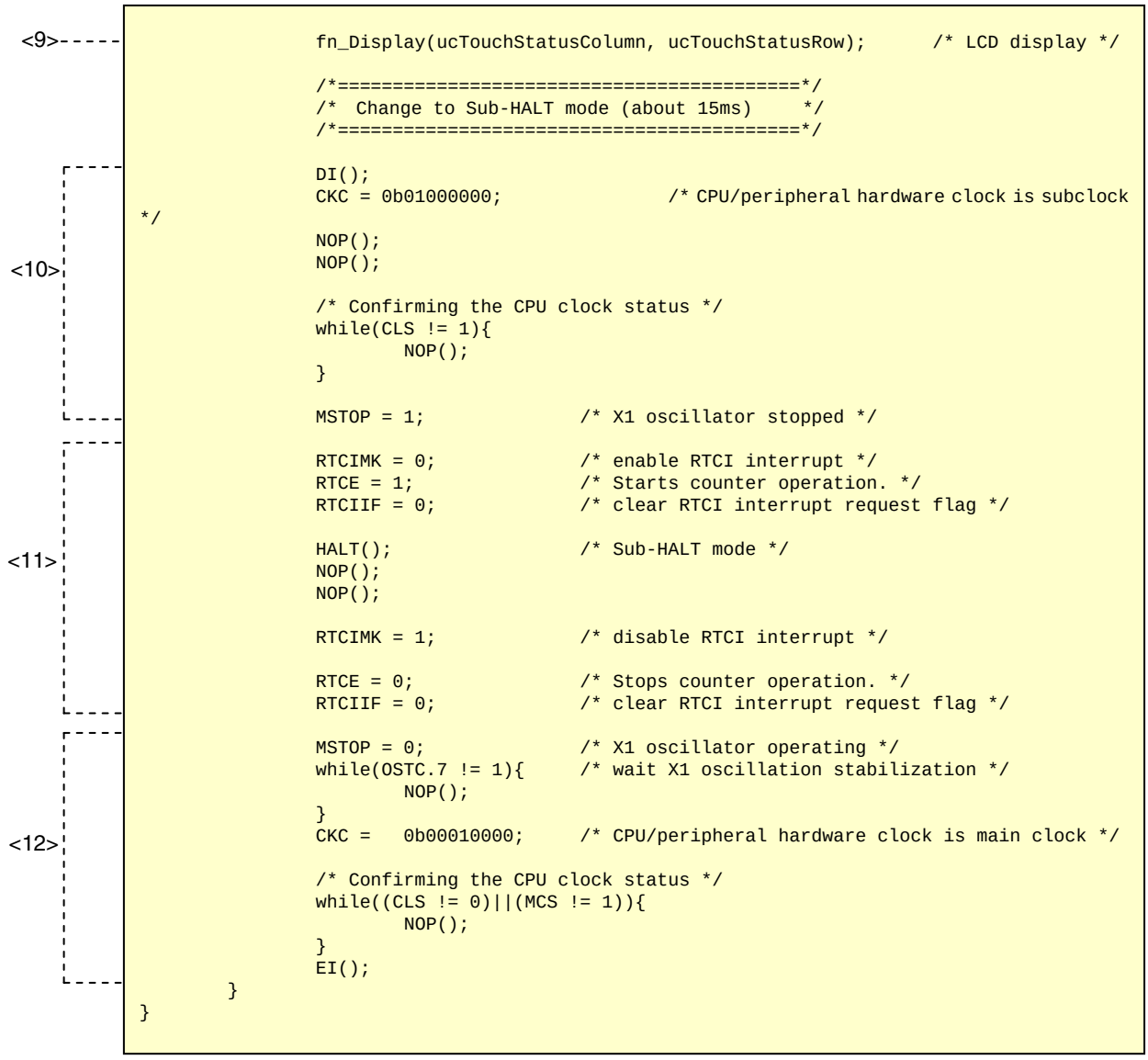
```
void main(void)
{

        unsigned char   ucCount;                /* work counter */


        for(ucCount = 0; ucCount<4 ; ucCount++){
                ushColumnOff[ucCount] = 0xffff;                 /* Column Notouch data */
                ushColumnCurrent[ucCount] = 0xffff;             /* Column current data */
        }

        for(ucCount = 0; ucCount<2 ; ucCount++){
                ushRowOff[ucCount] = 0xffff;                    /* Row Notouch data */
                ushRowCurrent[ucCount] = 0xffff;                /* Row current data */
        }

        ucTouchStatusColumnWork = 0;            /* touch column position work */
        ucTouchStatusRowWork = 0;               /* touch row position work */

        ucTouchStatusColumnLast = 0;            /* last touch column position work */
        ucTouchStatusRowLast = 0;               /* last touch row position work */

        ucTouchStatusColumn = 0;                /* touch column position */
        ucTouchStatusRow = 0;                   /* touch row position */

<1>     EGP0 |= 0b10111110;             /* set INTP1-INTP5,INTP7 to Rising edge */
        EGN0 &= 0b01000001;             /* reset INTP1-INTP5,INTP7 to Falling edge */

        /*==================================*/
        /*----------------------------------*/
        /*          Main Loop               */
        /*----------------------------------*/
        /*==================================*/

<2>     fn_Display(ucTouchStatusRow, ucTouchStatusColumn);              /* initial display
*/

        /*================================================================*/
        /* if system have nothing to do, go to standby for power-saving */
        /*================================================================*/

        while (1){

                /*======================*/
                /* check P3.0(INTP1)    */
                /*======================*/

                P3.0 = 0;                       /* P3.0(INTP1) Latch Low */
(a)             PM3.0 = 0;                      /* set P3.0 output to discharge */
                fn_Wait500usBase(500/500);      /* wait discharge */

<3>             PIF1 = 0;                       /* clear INTP1 request */
                TS0L.6 = 1;                     /* start TAU0 CH6 */
(b)             TMIF06 = 0;                     /* clear TM6 interrupt request */
                PM3.0 = 1;                      /* set P3.0 input to start charge */
                while(PIF1 == 0);               /* wait charge end */
                ushColumnCurrent[0] = TCR06;/* get timer count */
                if(ushColumnOff[0] > ushColumnCurrent[0]){
(c)                     ushColumnOff[0] = ushColumnCurrent[0];  /* set touch pad off data */
                }
                TT0L.6 = 1;                                     /* stop TAU0 CH6 */
```

```
/*=========================*/
/* check P3.1(INTP2)       */
/*=========================*/

P3.1 = 0;                       /* P3.1(INTP2) Latch Low */
PM3.1 = 0;                      /* set P3.1 output to discharge */
fn_Wait500usBase(500/500);      /* wait discharge */

PIF2 = 0;                       /* clear INTP2 request */
TS0L.6 = 1;                     /* start TAU0 CH6 */
TMIF06 = 0;                     /* clear TM6 interrupt request */
PM3.1 = 1;                      /* set P3.1 input to start charge */
while(PIF2 == 0);               /* wait charge end */
ushColumnCurrent[1] = TCR06;/* get timer count */
if(ushColumnOff[1] > ushColumnCurrent[1]){
        ushColumnOff[1] = ushColumnCurrent[1];  /* set touch pad off data */
}
TT0L.6 = 1;                                     /* stop TAU0 CH6 */

/*=========================*/
/* check P3.2(INTP5)       */
/*=========================*/

P3.2 = 0;                       /* P3.2(INTP5) Latch Low */
PM3.2 = 0;                      /* set P3.2 output to discharge */
fn_Wait500usBase(500/500);      /* wait discharge */

PIF5 = 0;                       /* clear INTP5 request */
TS0L.6 = 1;                     /* start TAU0 CH6 */
TMIF06 = 0;                     /* clear TM6 interrupt request */
PM3.2 = 1;                      /* set P3.2 input to start charge */
while(PIF5 == 0);               /* wait charge end */
ushColumnCurrent[2] = TCR06;/* get timer count */
if(ushColumnOff[2] > ushColumnCurrent[2]){
        ushColumnOff[2] = ushColumnCurrent[2];  /* set touch pad off data */
}
TT0L.6 = 1;                                     /* stop TAU0 CH6 */
```

<3>

```
/*=========================*/
/* check P3.3(INTP3)       */
/*=========================*/

P3.3 = 0;                       /* P3.3(INTP3) Latch Low */
PM3.3 = 0;                      /* set P3.3 output to discharge */
fn_Wait500usBase(500/500);      /* wait discharge */

PIF3 = 0;                       /* clear INTP3 request */
TS0L.6 = 1;                     /* start TAU0 CH6 */
TMIF06 = 0;                     /* clear TM6 interrupt request */
PM3.3 = 1;                      /* set P3.3 input to start charge */
while(PIF3 == 0);               /* wait charge end */
ushColumnCurrent[3] = TCR06;/* get timer count */
if(ushColumnOff[3] > ushColumnCurrent[3]){
        ushColumnOff[3] = ushColumnCurrent[3];  /* set touch pad off data */
}
TT0L.6 = 1;                                     /* stop TAU0 CH6 */


/*=========================*/
/* check P1.4(INTP4)       */
/*=========================*/

P1.4 = 0;                       /* P1.4(INTP4) Latch Low */
PM1.4 = 0;                      /* set P1.4 output to discharge */
fn_Wait500usBase(500/500);      /* wait discharge */
PIF4 = 0;                       /* clear INTP4 request */
TS0L.6 = 1;                     /* start TAU0 CH6 */
TMIF06 = 0;                     /* clear TM6 interrupt request */
PM1.4 = 1;                      /* set P1.4 input to start charge */
while(PIF4 == 0);               /* wait charge end */
ushRowCurrent[0] = TCR06;       /* get timer count */
if(ushRowOff[0] > ushRowCurrent[0]){
        ushRowOff[0] = ushRowCurrent[0];        /* set touch pad off data */
}
TT0L.6 = 1;                             /* stop TAU0 CH6 */
```

```
                /*=========================*/
                /* check P1.5(INTP7)       */
                /*=========================*/

                P1.5 = 0;                       /* P1.5(INTP7) Latch Low */
                PM1.5 = 0;                      /* set P1.5 output to discharge */
                fn_Wait500usBase(500/500);      /* wait discharge */
<3>
                PIF7 = 0;                       /* clear INTP7 request */
                TS0L.6 = 1;                     /* start TAU0 CH6 */
                TMIF06 = 0;                     /* clear TM6 interrupt request */
                PM1.5 = 1;                      /* set P1.5 input to start charge */
                while(PIF7 == 0);               /* wait charge end */
                ushRowCurrent[1] = TCR06;       /* get timer count */
                if(ushRowOff[1] > ushRowCurrent[1]){
                        ushRowOff[1] = ushRowCurrent[1];        /* set touch pad off data */
                }
                TT0L.6 = 1;                             /* stop TAU0 CH6 */

                /*=========================*/
                /* check touch position    */
                /*=========================*/

                ucTouchStatusColumnWork = 0;            /* touch column position work */
                ucTouchStatusColumn = 0;                /* touch column position */

                for(ucCount = 0; ucCount < 4; ucCount++){
                        if((ushColumnCurrent[ucCount] - ushColumnOff[ucCount]) > 5){
                                                /* touch detect in column */
                                if(ucTouchStatusColumnWork != 0){
                                        /* multi position touch detect */
<4>                                     ucTouchStatusColumnWork = 0xff;
                                        break;
                                }
                                else{
                                        ucTouchStatusColumnWork  =  ucCount;/*  set  touch
position */
                                }
                        }
                }

                for(ucCount = 0; ucCount < 2; ucCount++){
                        if((ushRowCurrent[ucCount] - ushRowOff[ucCount]) > 5){
                                                /* touch detect in row */
                                if(ucTouchStatusRowWork != 0){
                                        /* multi position touch detect */
                                        ucTouchStatusRowWork = 0xff;
<5>                                     break;
                                }
                                else{
                                        ucTouchStatusRowWork = ucCount;/* set touch position
*/
                                }
                        }
                }

                        /* compare last detect position to remove noize */
                if((ucTouchStatusColumnLast == ucTouchStatusColumnWork) &&
                        (ucTouchStatusRowLast == ucTouchStatusRowWork)){
<6>                     ucTouchStatusColumn = ucTouchStatusColumnWork;
                        ucTouchStatusRow = ucTouchStatusRowWork;
                }
                        /* renew last status */
                ucTouchStatusColumnLast = ucTouchStatusColumnWork;
                ucTouchStatusRowLast = ucTouchStatusRowWork;

                        /* notouch point adjust */
                if((ucTouchStatusColumn == 0x00)||(ucTouchStatusRow == 0x00)){
<7>                     ucTouchStatusColumn = 0x00;
                        ucTouchStatusRow = 0x00;
                }

                        /* multi touch point adjust */
                if((ucTouchStatusColumn == 0xff)||(ucTouchStatusRow == 0xff)){
<8>                     ucTouchStatusColumn = 0xff;
                        ucTouchStatusRow = 0xff;
                }
```



```
                fn_Display(ucTouchStatusColumn, ucTouchStatusRow);      /* LCD display */

                /*=========================================*/
                /*  Change to Sub-HALT mode (about 15ms)    */
                /*=========================================*/

                DI();
                CKC = 0b01000000;               /* CPU/peripheral hardware clock is subclock
*/
                NOP();
                NOP();

                /* Confirming the CPU clock status */
                while(CLS != 1){
                        NOP();
                }

                MSTOP = 1;              /* X1 oscillator stopped */

                RTCIMK = 0;             /* enable RTCI interrupt */
                RTCE = 1;               /* Starts counter operation. */
                RTCIIF = 0;             /* clear RTCI interrupt request flag */

                HALT();                 /* Sub-HALT mode */
                NOP();
                NOP();

                RTCIMK = 1;             /* disable RTCI interrupt */

                RTCE = 0;               /* Stops counter operation. */
                RTCIIF = 0;             /* clear RTCI interrupt request flag */

                MSTOP = 0;              /* X1 oscillator operating */
                while(OSTC.7 != 1){     /* wait X1 oscillation stabilization */
                        NOP();
                }
                CKC =   0b00010000;     /* CPU/peripheral hardware clock is main clock */

                /* Confirming the CPU clock status */
                while((CLS != 0)||(MCS != 1)){
                        NOP();
                }
                EI();
        }
}
```

CHAPTER 6 EXAMPLE OF OPERATION CHECK USING DEVICE

This chapter illustrates how to check the operation of the touch sensor.

(1) When the touch sensor is not touched

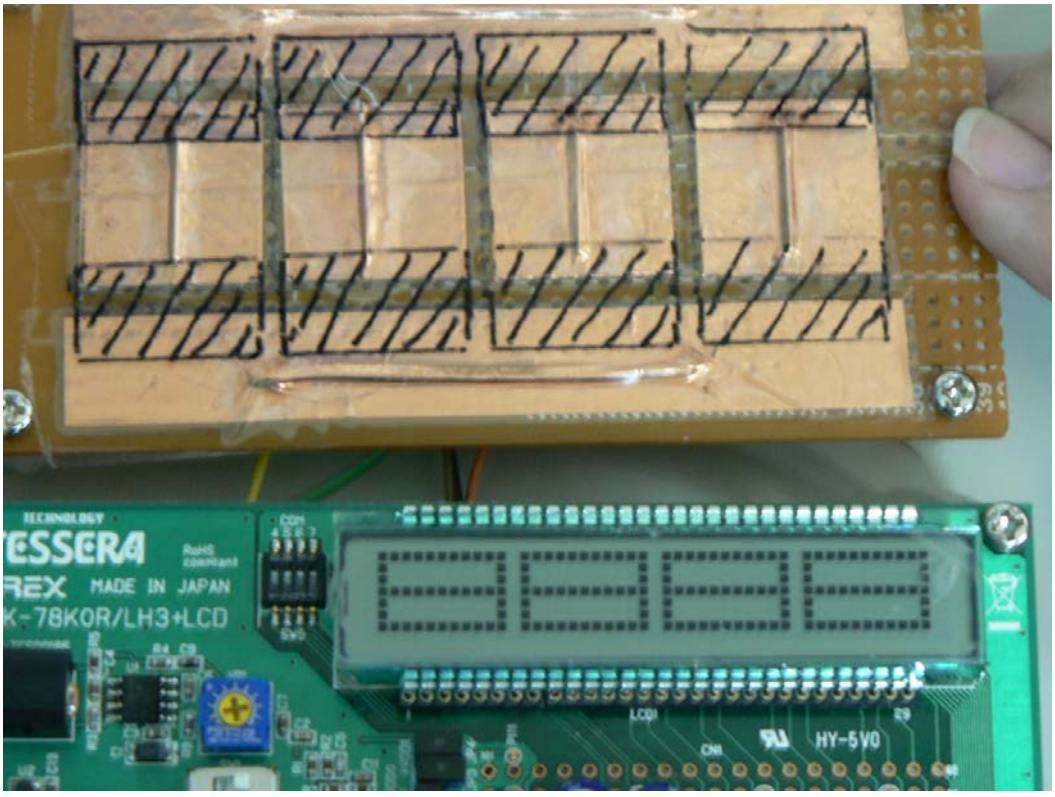


Figure 6-1. Results of Checking Operation When Touch Sensor Is Not Touched

(2) When the touch sensor is touched

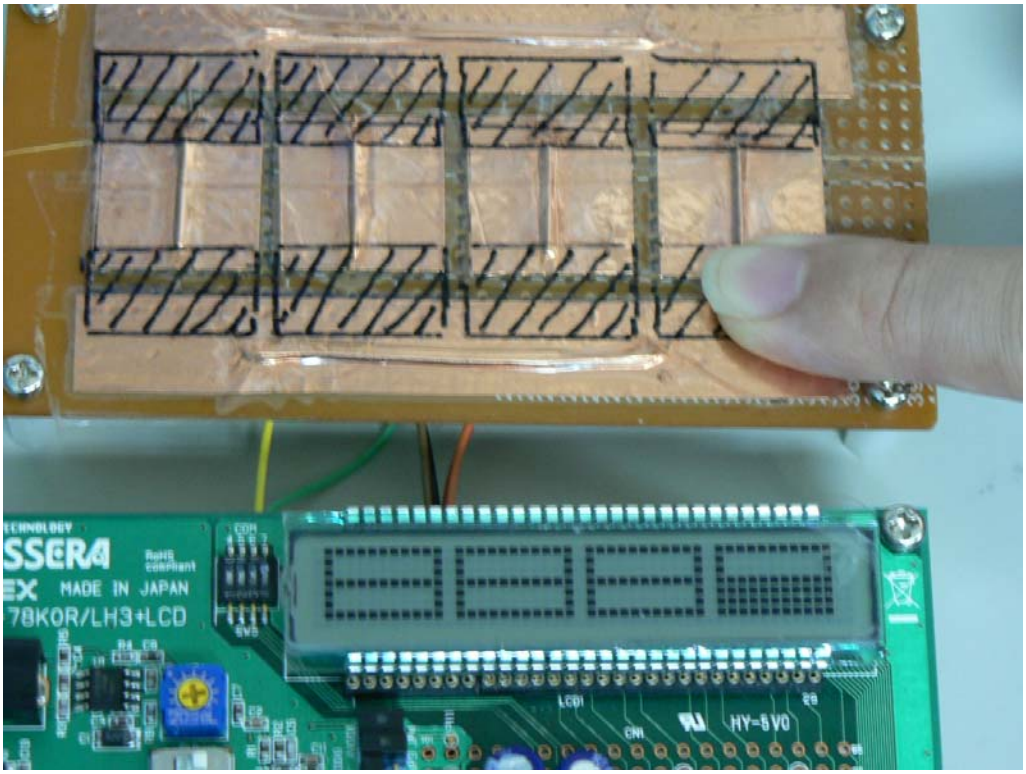


Figure 6-2. Results of Checking Operation When Touch Sensor Is Touched (1)

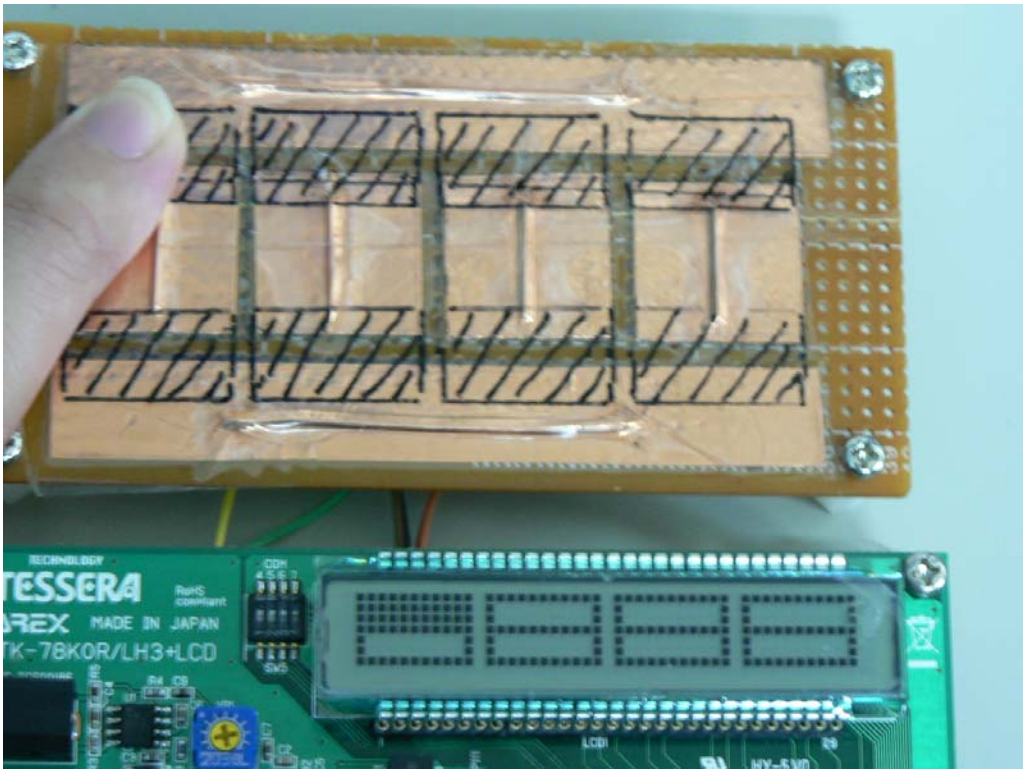


Figure 6-3. Results of Checking Operation When Touch Sensor Is Touched (2)

(3) When touch sensor is touched in more than one place at a time

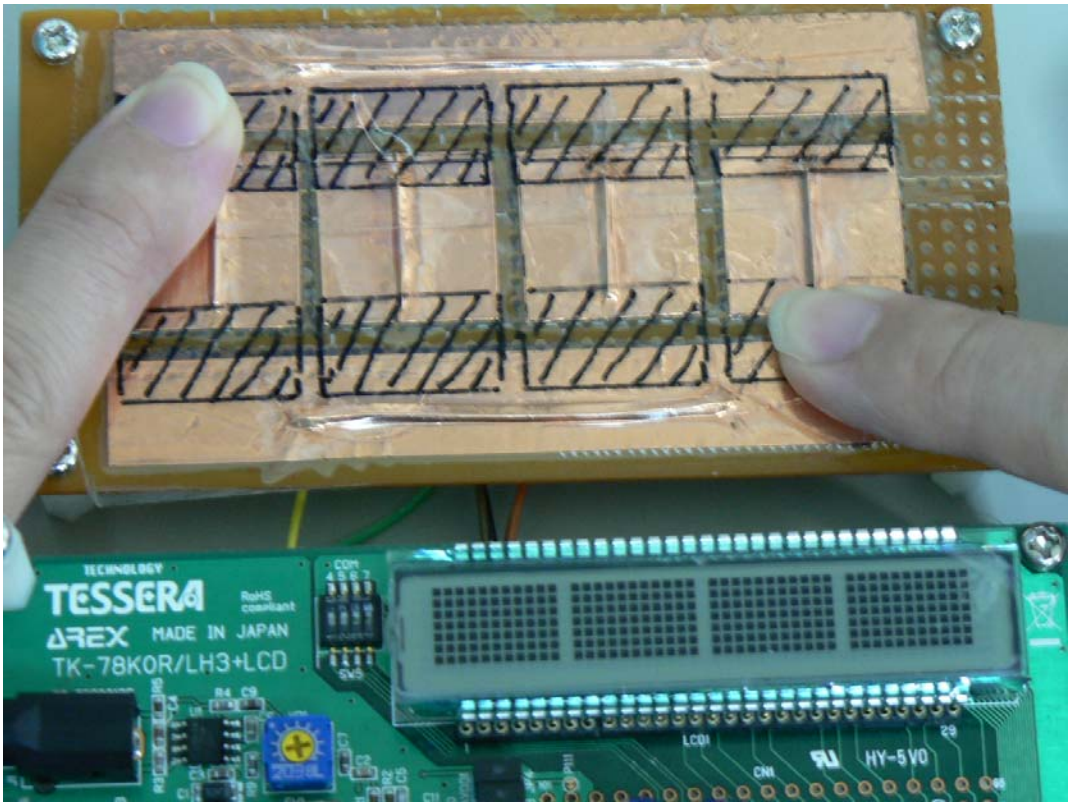


Figure 6-4. Results of Checking Operation When Touch Sensor Is Touched in More Than One Place at a Time

CHAPTER 7 RELATED DOCUMENTS

Document Name		English
78K0R/Lx3 User's Manual		PDF
TK-78K0R/LH3+LCD User's Manual		PDF
CC78K0R C Compiler User's Manual	Language	PDF
	Operation	PDF
PM+ Project Manager User's Manual		PDF

Remark TK-78K0R/LH3+LCD is a product of TESSERA TECHNOLOGY INC.
Contact: TESSERA TECHNOLOGY INC. (http://www.tessera.co.jp)

APPENDIX A PROGRAM LIST

As a program list example, the source program is shown below.

● main.c

```
/*
 * Copyright (C) NEC Electronics Corporation 2006
 * NEC ELECTRONICS CONFIDENTIAL AND PROPRIETARY
 * All rights reserved by NEC Electronics Corporation.
 * This program must be used solely for the purpose for which
 * it was furnished by NEC Electronics Corporation. No part of this
 * program may be reproduced or disclosed to others, in any
 * form, without the prior written permission of NEC Electronics
 * Corporation. Use of copyright notice dose not evidence
 * publication of the program.
 */

/*------------------------------------------------------------------------*/
/* #pragma directive for CC78K0                                           */
/*------------------------------------------------------------------------*/
#pragma         SFR
#pragma         DI
#pragma         EI
#pragma         HALT
#pragma         NOP


/*------------------------------------------------------------------------*/
/* Include files                                                          */
/*------------------------------------------------------------------------*/
/* TAU:TDR0n value operation by CK00 (fCLK/2^3 = 5MHz) */
#define CCK00_500USEC    (2500 - 1)             /* 500us (0.2[us/clk] * 2500[count]) */


/*------------------------------------------------------------------------*/
/* Function prototyps
```

```
                                        */
    /*---------------------------------------------------------------------*/
                    void    fn_Wait500usBase(unsigned short);       /* Delays the program for (Time * 500us) */
                    void    fn_InitPort(void);                                  /* Setting of I/O ports
*/
                    void    fn_InitTau0(void);                                  /* Setting of Timer
array unit 0 */
                    void    fn_InitLvi(void);                                   /* Setting of
Low-voltage detectior */
                    void    fn_InitRtc(void);                                   /* Setting of
Real-time counter */

    extern  void    fn_DisplayInit(void);                               /* Setting of LCD driver */
    extern  void    fn_Display(unsigned char ucColumn, unsigned char ucRow);    /* LCD display */

    /*---------------------------------------------------------------------*/
    /* Extern variables/constants
                    */
    /*---------------------------------------------------------------------*/

    /*---------------------------------------------------------------------*/
    /* Local constants
                                        */
    /*---------------------------------------------------------------------*/

    /*---------------------------------------------------------------------*/
    /* Global variables
                                        */
    /*---------------------------------------------------------------------*/

    /*---------------------------------------------------------------------*/
    /* Local variables
                                        */
    /*---------------------------------------------------------------------*/

    static  unsigned short      ushColumnOff[4];    /* Column Notouch data */
    static  unsigned short      ushRowOff[2];               /* Row Notouch data */

    static  unsigned short      ushColumnCurrent[4];        /* Column current data */
```

```
static   unsigned short      ushRowCurrent[2];              /* Row current data */

static   unsigned char       ucTouchStatusColumn;           /* touch column position */
static   unsigned char       ucTouchStatusRow;              /* touch row position */

static   unsigned char       ucTouchStatusColumnWork;       /* touch column position work */
static   unsigned char       ucTouchStatusRowWork;                  /* touch row position work */

static   unsigned char       ucTouchStatusColumnLast;       /* last touch column position work */
static   unsigned char       ucTouchStatusRowLast;                  /* last touch row position work */


/*---------------------------------------------------------------------------*/
/* Code                                                                      */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Hardware initialization                                                   */
/*---------------------------------------------------------------------------*/
void hdwinit(void)
{
        DI();                                   /* disable all interrupts */

        /*-----------------------------------------*/
        /*          Initialization of port         */
        /*-----------------------------------------*/
        fn_InitPort();

        /*-----------------------------------------*/
        /*          Initialization of clock        */
        /*-----------------------------------------*/
        CMC =    0b01010011;                    /* Clock Operation Mode Control Register */
                                /*|||||||+--- : Control of high-speed system clock oscillation frequency */
                                /*|||||||  : 0 : 2 MHz <= fMX <= 10 MHz */
                                /*|||||||  : 1 : 10 MHz < fMX <= 20 MHz */
                                /*|||||||  */
                                /*|||||++---- : XT1 oscillator oscillation mode selection */
```

```
                    /*|||||  : 0 0 : Low-consumption oscillation */
                    /*|||||  : 0 1 : Normal oscillation */
                    /*|||||  : 1 x : Super-low-consumption oscillation */
                    /*|||||     x = don't care */
                    /*||||| */
                    /*||||+------ : Be sure to set 0 */
                    /*|||| */
                    /*|||+------- : [1] Subsystem clock pin operation mode */
                    /*|||                [2] XT1/P123 pin and XT2/P124 pin */
                    /*|||  : 0 : [1]Input port mode */
                    /*|||          [2]Input port */
                    /*||| */
                    /*|||  : 1 : [1]XT1 oscillation mode */
                    /*|||          [2]Crystal resonator connection */
                    /*||| */
                    /*||+-------- : Be sure to set 0 */
                    /*|| */
                    /*++--------- : [1]EXCLK OSCSEL High-speed system clock pin operation mode */
                    /*                    [2]X1/P121 pin */
                    /*                    [3]X2/EXCLK/P122 pin */
                    /*  : 0 0 : [1]Input port mode */
                    /*            [2][3]Input port */
                    /* */
                    /*  : 0 1 : [1]X1 oscillation mode */
                    /*            [2][3]Crystal/ceramic resonator connection */
                    /* */
                    /*  : 1 0 : [1]Input port mode */
                    /*            [2][3]Input port */
                    /* */
                    /*  : 1 1 : [1]External clock input mode */
                    /*            [2]Input port */
                    /*            [3]External clock input */

MSTOP = 0;                          /* X1 oscillator operating */
XTSTOP = 0;                         /* XT1 oscillator operating */

OSMC =  0b00000001;                 /* Operation Speed Mode Control Register */
                    /*||||||++--- : fCLK frequency selection */
                    /*||||||  : 0 0 : Operates at a frequency of 10 MHz or less. */
```

```
                        /*||||||  : 0 1 : Operates at a frequency higher than 10 MHz. */
                        /*||||||  : 1 0 : Operates at a frequency of 1 MHz. */
                        /*||||||  */
                        /*|+++++----- : Be sure to set 00000 */
                        /*|  */
                        /*+---------- : Setting in subsystem clock HALT mode */
                        /*  : 0 : Enables subsystem clock supply to peripheral functions. */
                        /*          (See Table 21-1 Operating Statuses in HALT Mode (2/3) */
                        /*           for the peripheral functions whose operations are enabled.) */
                        /*  : 1 : Stops subsystem clock supply to peripheral functions except real-time counter,
*/
                        /*          clock output/buzzer output, and LCD controller/driver. */

    while(OSTC.0 != 1){             /* wait X1 oscillation stabilization */
            NOP();
    }

    /*-- Caution ------------------------------------------------*/
    /* To increase fCLK to 10 MHz or higher, set FSEL to '1',    */
    /* then change fCLK after two or more clocks have elapsed. */
    /*-------------------------------------------------------------*/
    NOP();
    NOP();

    CKC =   0b00010000;                     /* System Clock Control Register */
                        /*|+|+++++--- : Selection of CPU/peripheral hardware clock (fCLK) */
                        /*| |  : 0 0 x 0 0 0 : fIH */
                        /*| |  : 0 0 x 0 0 1 : fIH/2 (default) */
                        /*| |  : 0 0 x 0 1 0 : fIH/2^2 */
                        /*| |  : 0 0 x 0 1 1 : fIH/2^3 */
                        /*| |  : 0 0 x 1 0 0 : fIH/2^4 */
                        /*| |  : 0 0 x 1 0 1 : fIH/2^5 */
                        /*| |  : 0 1 x 0 0 0 : fMX */
                        /*| |  : 0 1 x 0 0 1 : fMX/2 */
                        /*| |  : 0 1 x 0 1 0 : fMX/2^2 */
                        /*| |  : 0 1 x 0 1 1 : fMX/2^3 */
                        /*| |  : 0 1 x 1 0 0 : fMX/2^4 */
                        /*| |  : 0 1 x 1 0 1 : fMX/2^5 */
                        /*| |  : 1 x 0 x x x : fSUB */
```

```
                        /*| | : 1 x 1 x x x : fSUB/2 */
                        /*| | : Other than above : Setting prohibited */
                        /*| |    x = don't care */
                        /*| | */
                        /*| +-------- : Status of Main system clock (fMAIN) */
                        /*| : 0 : Internal high-speed oscillation clock (fIH) */
                        /*| : 1 : High-speed system clock (fMX) */
                        /*| */
                        /*+---------- : Status of CPU/peripheral hardware clock (fCLK) */
                        /* : 0 : Main system clock (fMAIN) */
                        /* : 1 : Subsystem clock (fSUB) */

/* Confirming the CPU clock status */
while((CLS != 0)||(MCS != 1)){
            NOP();
}
/* CPU is operating on a High-speed system clock */
HIOSTOP = 1;                            /* internal high-speed oscillation stopped */

OSTS =    0b00000111;                   /* Oscillation Stabilization Time Select Register */
                        /*||||++++--- : Oscillation stabilization time selection */
                        /*|||| : 0 0 0 : 2^8/fX */
                        /*|||| : 0 0 1 : 2^9/fX */
                        /*|||| : 0 1 0 : 2^10/fX */
                        /*|||| : 0 1 1 : 2^11/fX */
                        /*|||| : 1 0 0 : 2^13/fX */
                        /*|||| : 1 0 1 : 2^15/fX */
                        /*|||| : 1 1 0 : 2^17/fX */
                        /*|||| : 1 1 1 : 2^18/fX */
                        /*|||| */
                        /*++++------- : Be sure to set 000000 */

/*---------------------------------------*/
/*            Initialization of timer            */
/*---------------------------------------*/
fn_InitTau0();

/* software to wait for the operation stabilization time */
/* (over 200ms from when XT1 enable) */
```

```
    fn_Wait500usBase(200000/500);

    /*---------------------------------------*/
    /* Initialization of low-voltage detector */
    /*---------------------------------------*/
    fn_InitLvi();

    /*---------------------------------------*/
    /*    Initialization of real-time counter     */
    /*---------------------------------------*/
    fn_InitRtc();

    /*---------------------------------------*/
    /*    Initialization of LCD                                          */
    /*---------------------------------------*/
    fn_DisplayInit();


    EI();                                   /* enable all interrupts */

}

/*-------------------------------------------------------------------*/
/* Module:          fn_InitPort
                    */
/* Description:     Setting of I/O ports                                                    */
/*       parameter: --
                                */
/*       return     : --
                                */
/*-------------------------------------------------------------------*/
void    fn_InitPort(void)
{
/*-----------------------------------------------------------*/
/*       Ports configuration for digital input and output                  */
/*-----------------------------------------------------------*/
    ADPC =   0b00010000;        /* A/D Port Configuration Register */
                        /*|||+++++--- : Analog input (A)/digital I/O (D) switching */
                        /*||| :                +------------------------- ANI15/AVREFM/P157 */
```

```
                        /*lll :                 | +-+-+------------------- ANI10/P152 - ANI8/AMP2+/P150 */
                        /*lll :                 | | | | +-+-+-+-+-+-+-+-+--- ANI7/AMP2O/P27 - ANI0/AMP0-/P20 */
                        /*lll : 0 0 0 0 0 : A A A A A A A A A A A A */
                        /*lll : 0 0 0 0 1 : A A A A A A A A A A A D */
                        /*lll : 0 0 0 1 0 : A A A A A A A A A A A D D */
                        /*lll : 0 0 0 1 1 : A A A A A A A A A A D D D */
                        /*lll : 0 0 1 0 0 : A A A A A A A A A D D D D */
                        /*lll : 0 0 1 0 1 : A A A A A A A A D D D D D */
                        /*lll : 0 0 1 1 0 : A A A A A A A D D D D D D */
                        /*lll : 0 0 1 1 1 : A A A A A A D D D D D D D */
                        /*lll : 0 1 0 0 0 : A A A A D D D D D D D D */
                        /*lll : 0 1 0 0 1 : A A A D D D D D D D D D */
                        /*lll : 0 1 0 1 0 : A A D D D D D D D D D D */
                        /*lll : 0 1 1 1 1 : A D D D D D D D D D D D */
                        /*lll : 1 0 0 0 0 : D D D D D D D D D D D D */
                        /*lll */
                        /*+++-------- : Be sure to set 000 */

/*-------------------------------------------------------------*/
/*      Setting of Port 0
      */
/*-------------------------------------------------------------*/
      P0 =    0b00000000;       /* Set P00-P02 Output latch to Low */
      PM0 =   0b11111000;       /* Set P00-P02 to output port */
                                                  /* P00-P02:Unused */

/*-------------------------------------------------------------*/
/*      Setting of Port 1
      */
/*-------------------------------------------------------------*/
      P1 =    0b00000000;       /* Set P10-P17 Output latch to Low */
      PM1 =   0b00000000;       /* Set P10-P17 to output port */
                                                  /* P10-P13:Unused */
      PU1 =   0b00110000;       /* P14,P15 to On-chip pull-up resistor connect */

/*-------------------------------------------------------------*/
/*      Setting of Port 2
      */
/*-------------------------------------------------------------*/
```

```
        P2 =    0b00000000;     /* Set P20-P27 Output latch to Low */
        PM2 =   0b11111111;     /* Set P20-P27 to input port */
                                                /* P20-P27:Unused */

/*-------------------------------------------------------------*/
/*      Setting of Port 3
        */
/*-------------------------------------------------------------*/
        P3 =    0b00000000;     /* Set P30-P33 Output latch to Low */
                                                /* Set P33,P32 Output latch High */
        PM3 =   0b11110000;     /* Set P30-P33 to output port */
                                                /* Set P34(TI06) to input port */
        PU3 =   0b00001111;     /* P30-P33 to On-chip pull-up resistor connect */

/*-------------------------------------------------------------*/
/*      Setting of Port 4
        */
/*-------------------------------------------------------------*/
        P4 =    0b00000000;     /* Set P40-P41 Output latch to Low */
        PM4 =   0b11111100;     /* Set P40-P41 to output port */
                                                /* P40-P41:Unused */

/*-------------------------------------------------------------*/
/*      Setting of Port 5
        */
/*-------------------------------------------------------------*/
        P5 =    0b00000000;     /* Set P50-P57 Output latch to Low */
        PM5 =   0b11110000;     /* Set P50-P57 to output port */
                                                /* P50-P57:Unused */

/*-------------------------------------------------------------*/
/*      Setting of Port 6
        */
/*-------------------------------------------------------------*/
        P6 =    0b00000000;     /* Set P60-P61 Output latch to Low */
        PM6 =   0b11111100;     /* Set P60-P61 to output port */
                                                /* P60-P61:Unused */

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

```
/*      Setting of Port 7
        */
/*------------------------------------------------------------*/
        P7 =    0b00000000;         /* Set P70-P77 Output latch to Low */
        PM7 =   0b00000000;         /* Set P70-P77 to output port */
                                                    /* P70-P77:Unused */

/*------------------------------------------------------------*/
/*      Setting of Port 8
        */
/*------------------------------------------------------------*/
        P8 =    0b00000000;         /* Set P80-P88 Output latch to Low */
        PM8 =   0b00000000;         /* Set P80-P88 to output port */
                                                    /* P80-P88:Unused */

/*------------------------------------------------------------*/
/*      Setting of Port 9
        */
/*------------------------------------------------------------*/
        P9 =    0b00000000;         /* Set P90-P97 Output latch to Low */
        PM9 =   0b00000000;         /* Set P90-P97 to output port */
                                                    /* P90-P97:Unused */

/*------------------------------------------------------------*/
/*      Setting of Port 10
        */
/*------------------------------------------------------------*/
        P10 =   0b00000000;         /* Set P100-P102 Output latch to Low */
        PM10 =  0b11111000;         /* Set P100-P102 to output port */
                                                    /* P100-P102:Unused */

/*------------------------------------------------------------*/
/*      Setting of Port 11
        */
/*------------------------------------------------------------*/
        P11 =   0b00000000;         /* Set P110-P111 Output latch to Low */
        PM11 =  0b11111100;         /* Set P110-P111 to output port */
                                                    /* P110-P111:Unused */
```

```
/*-------------------------------------------------------------*/
/*      Setting of Port 12
        */
/*-------------------------------------------------------------*/
        P12 =   0b00000000;         /* Set P120 Output latch to Low */
        PM12 =  0b11111111;         /* Set P120 to output port */
                                                /* P121-P124:Unused */
                                                /* *P120-P124:Input port */

/*-------------------------------------------------------------*/
/*      Setting of Port 13
        */
/*-------------------------------------------------------------*/
        P13 =   0b00000000;         /* Set P130 Output latch to Low */
                                                /* P130:Unused */

/*-------------------------------------------------------------*/
/*      Setting of Port 14
        */
/*-------------------------------------------------------------*/
        P14 =   0b00000000;         /* Set P140-P147 Output latch to Low */
        PM14 =  0b00000000;         /* Set P140-P147 to output port */
                                                /* P140-P147:Unused */

/*-------------------------------------------------------------*/
/*      Setting of Port 15
        */
/*-------------------------------------------------------------*/
        P15 =   0b00000000;         /* Set P150-P152,P157 Output latch to Low */
        PM15 =  0b11111111;         /* Set P150-P152,P157 to input port */
                                                /* P150-P152,P157:Unused */

}

/*-------------------------------------------------------------------*/
/* Module:          fn_InitTau0
                                */
/* Description:     Setting of Timer array unit 0                                   */
/*      parameter: --
```

```
                                */
/*      return     : --
                                */
/*-------------------------------------------------------------------*/
void    fn_InitTau0(void)
{
        TAU0EN = 1;                             /* supplies input clock to timer array unit 0 */
        TPS0L =   0b00000010;           /* Timer Clock Select Register 0 */
                                /*||||++++--- : Selection of operation clock (CK00) */
                                /*++++------- : Selection of operation clock (CK01) */
                                /* : 0 0 0 0 : CK0m = fCLK */
                                /* : 0 0 0 1 : CK0m = fCLK/2 */
                                /* : 0 0 1 0 : CK0m = fCLK/2^2 */
                                /* : 0 0 1 1 : CK0m = fCLK/2^3 */
                                /* : 0 1 0 0 : CK0m = fCLK/2^4 */
                                /* : 0 1 0 1 : CK0m = fCLK/2^5 */
                                /* : 0 1 1 0 : CK0m = fCLK/2^6 */
                                /* : 0 1 1 1 : CK0m = fCLK/2^7 */
                                /* : 1 0 0 0 : CK0m = fCLK/2^8 */
                                /* : 1 0 0 1 : CK0m = fCLK/2^9 */
                                /* : 1 0 1 0 : CK0m = fCLK/2^10 */
                                /* : 1 0 1 1 : CK0m = fCLK/2^11 */
                                /* : 1 1 0 0 : CK0m = fCLK/2^12 */
                                /* : 1 1 0 1 : CK0m = fCLK/2^13 */
                                /* : 1 1 1 0 : CK0m = fCLK/2^14 */
                                /* : 1 1 1 1 : CK0m = fCLK/2^15 */
                                /*    m = 0, 1 */

        /* CH1:for wait */
        TMR01 = 0b0000000000000000;             /* Timer Mode Register 01 */
                                /*||||||||||||++++--- : [1]Operation mode of channel 1 */
                                /*||||||||||||               [2]Count operation of TCR */
                                /*||||||||||||               [3]Independent operation */
                                /*||||||||||||               [4]Setting of starting counting and interrupt */
                                /*||||||||||||  : 0 0 0 0 : [1]Interval timer mode */
                                /*||||||||||||                   [2]Counting down */
                                /*||||||||||||                   [3]Possible */
                                /*||||||||||||                   [4]Timer interrupt is not generated when counting is started */
                                /*||||||||||||                       (timer output does not change, either). */
```

```
/*||||||||||||| */
/*||||||||||||| : 0 0 0 1 : [1]Interval timer mode */
/*|||||||||||||                [2]Counting down */
/*|||||||||||||                [3]Possible */
/*|||||||||||||                [4]Timer interrupt is generated when counting is started */
/*|||||||||||||                    (timer output also changes). */
/*||||||||||||| */
/*||||||||||||| : 0 1 0 0 : [1]Capture mode */
/*|||||||||||||                [2]Counting up */
/*|||||||||||||                [3]Possible */
/*|||||||||||||                [4]Timer interrupt is not generated when counting is started */
/*|||||||||||||                    (timer output does not change, either). */
/*||||||||||||| */
/*||||||||||||| : 0 1 0 1 : [1]Capture mode */
/*|||||||||||||                [2]Counting up */
/*|||||||||||||                [3]Possible */
/*|||||||||||||                [4]Timer interrupt is generated when counting is started */
/*|||||||||||||                    (timer output also changes). */
/*||||||||||||| */
/*||||||||||||| : 0 1 1 0 : [1]Event counter mode */
/*|||||||||||||                [2]Counting down */
/*|||||||||||||                [3]Possible */
/*|||||||||||||                [4]Timer interrupt is not generated when counting is started */
/*|||||||||||||                    (timer output does not change, either). */
/*||||||||||||| */
/*||||||||||||| : 1 0 0 0 : [1]One-count mode */
/*|||||||||||||                [2]Counting down */
/*|||||||||||||                [3]Impossible */
/*|||||||||||||                [4]Start trigger is invalid during counting operation. */
/*|||||||||||||                    At that time, interrupt is not generated, either. */
/*||||||||||||| */
/*||||||||||||| : 1 0 0 1 : [1]One-count mode */
/*|||||||||||||                [2]Counting down */
/*|||||||||||||                [3]Impossible */
/*|||||||||||||                [4]Start trigger is valid during counting operation. */
/*|||||||||||||                    At that time, interrupt is also generated. */
/*||||||||||||| */
/*||||||||||||| : 1 1 0 0 : [1]Capture & one-count mode */
/*|||||||||||||                [2]Counting up */
```

```
                        /*||||||||||||                  [3]Possible */
                        /*||||||||||||                  [4]Timer interrupt is not generated when counting is started */
                        /*||||||||||||                      (timer output does not change, either). */
                        /*||||||||||||                      Start trigger is invalid during counting operation. */
                        /*||||||||||||                      At that time interrupt is not generated, either. */
                        /*|||||||||||| */
                        /*|||||||||||| : Other than above : Setting prohibited */
                        /*|||||||||||| */
                        /*||||||||||++-------- : Be sure to set 00 */
                        /*|||||||||| */
                        /*||||||||++---------- : Selection of TI01 pin input signal, fSUB/2, fSUB/4, or INTRTC1 valid
edge */
                        /*||||||||                      (the timer input used with channel 1 is selected by using TIS0
register). */
                        /*||||||||  : 0 0 : Falling edge */
                        /*||||||||  : 0 1 : Rising edge */
                        /*||||||||  : 1 0 : Both edges (when low-level width is measured) */
                        /*||||||||            Start trigger: Falling edge, Capture trigger: Rising edge */
                        /*||||||||  : 1 1 : Both edges (when high-level width is measured) */
                        /*||||||||            Start trigger: Rising edge, Capture trigger: Falling edge */
                        /*||||||||  */
                        /*|||||+++------------ : Setting of start trigger or capture trigger of channel 1 */
                        /*|||||  : 0 0 0 : Only software trigger start is valid (other trigger sources are unselected).
*/
                        /*|||||  : 0 0 1 : Valid edge of TI01 pin input signal, fSUB/2, fSUB/4, or INTRTC1 is used
as both the start trigger and capture trigger. */
                        /*|||||  : 0 1 0 : Both the edges of TI01 pin input signal, fSUB/2, fSUB/4, or INTRTC1 are
used as a start trigger and a capture trigger. */
                        /*|||||  : 1 0 0 : Interrupt signal of the master channel is used (when the channel is used
as a slave channel with the combination operation function). */
                        /*|||||  : Other than above : Setting prohibited */
                        /*|||||  */
                        /*||||+---------------- : Selection of slave/master of channel 1 */
                        /*||||  : 0 : Operates as slave channel with combination operation function. */
                        /*||||  : 1 : Operates as master channel with combination operation function. */
                        /*||||  */
                        /*|||+----------------- : Selection of count clock (TCLK) of channel 0 */
                        /*|||  : 0 : Operation clock MCK specified by CKS01 bit */
                        /*|||  : 1 : Valid edge of input signal input from TI01 pin, fSUB/2, fSUB/4, or INTRTC1 */
```

```
                        /*|||           (the timer input used with channel 1 is selected by using TIS0 register). */
                        /*||| */
                        /*|++---------------- : Be sure to set 00 */
                        /*| */
                        /*+------------------ : Selection of operation clock (MCK) of channel 1 */
                        /* : 0 : Operation clock CK00 set by TPS0 register */
                        /* : 1 : Operation clock CK01 set by TPS0 register */
TDR01 = CCK00_500USEC;                  /* set interval time to 500us */
TMMK01 = 1;                                                     /* disable interrupt */

/* CH6:for charge period measure */
TMR06 = 0b1000000000000100;             /* Timer Mode Register 06 */
                        /*||||||||||||++++--- : [1]Operation mode of channel 6 */
                        /*||||||||||||               [2]Count operation of TCR */
                        /*||||||||||||               [3]Independent operation */
                        /*||||||||||||               [4]Setting of starting counting and interrupt */
                        /*||||||||||||  : 0 0 0 0 : [1]Interval timer mode */
                        /*||||||||||||                  [2]Counting down */
                        /*||||||||||||                  [3]Possible */
                        /*||||||||||||                  [4]Timer interrupt is not generated when counting is started */
                        /*||||||||||||                      (timer output does not change, either). */
                        /*||||||||||||  */
                        /*||||||||||||  : 0 0 0 1 : [1]Interval timer mode */
                        /*||||||||||||                  [2]Counting down */
                        /*||||||||||||                  [3]Possible */
                        /*||||||||||||                  [4]Timer interrupt is generated when counting is started */
                        /*||||||||||||                      (timer output also changes). */
                        /*||||||||||||  */
                        /*||||||||||||  : 0 1 0 0 : [1]Capture mode */
                        /*||||||||||||                  [2]Counting up */
                        /*||||||||||||                  [3]Possible */
                        /*||||||||||||                  [4]Timer interrupt is not generated when counting is started */
                        /*||||||||||||                      (timer output does not change, either). */
                        /*||||||||||||  */
                        /*||||||||||||  : 0 1 0 1 : [1]Capture mode */
                        /*||||||||||||                  [2]Counting up */
                        /*||||||||||||                  [3]Possible */
                        /*||||||||||||                  [4]Timer interrupt is generated when counting is started */
                        /*||||||||||||                      (timer output also changes). */
```

```
                        /*||||||||||| */
                        /*||||||||||| : 0 1 1 0 : [1]Event counter mode */
                        /*|||||||||||                  [2]Counting down */
                        /*|||||||||||                  [3]Possible */
                        /*|||||||||||                  [4]Timer interrupt is not generated when counting is started */
                        /*|||||||||||                      (timer output does not change, either). */
                        /*||||||||||| */
                        /*||||||||||| : 1 0 0 0 : [1]One-count mode */
                        /*|||||||||||                  [2]Counting down */
                        /*|||||||||||                  [3]Impossible */
                        /*|||||||||||                  [4]Start trigger is invalid during counting operation. */
                        /*|||||||||||                      At that time, interrupt is not generated, either. */
                        /*||||||||||| */
                        /*||||||||||| : 1 0 0 1 : [1]One-count mode */
                        /*|||||||||||                  [2]Counting down */
                        /*|||||||||||                  [3]Impossible */
                        /*|||||||||||                  [4]Start trigger is valid during counting operation. */
                        /*|||||||||||                      At that time, interrupt is also generated. */
                        /*||||||||||| */
                        /*||||||||||| : 1 1 0 0 : [1]Capture & one-count mode */
                        /*|||||||||||                  [2]Counting up */
                        /*|||||||||||                  [3]Possible */
                        /*|||||||||||                  [4]Timer interrupt is not generated when counting is started */
                        /*|||||||||||                      (timer output does not change, either). */
                        /*|||||||||||                      Start trigger is invalid during counting operation. */
                        /*|||||||||||                      At that time interrupt is not generated, either. */
                        /*||||||||||| */
                        /*||||||||||| : Other than above : Setting prohibited */
                        /*||||||||||| */
                        /*||||||||||++-------- : Be sure to set 00 */
                        /*|||||||||| */
                        /*||||||||++---------- : Selection of TI06 pin input signal, fSUB/2, fSUB/4, or INTRTC1 valid
edge */
                        /*||||||||                    (the timer input used with channel 4 is selected by using TIS0
register). */
                        /*||||||||  : 0 0 : Falling edge */
                        /*||||||||  : 0 1 : Rising edge */
                        /*||||||||  : 1 0 : Both edges (when low-level width is measured) */
                        /*||||||||          Start trigger: Falling edge, Capture trigger: Rising edge */
```

```
                                /*||||||||| : 1 1 : Both edges (when high-level width is measured) */
                                /*|||||||||                Start trigger: Rising edge, Capture trigger: Falling edge */
                                /*||||||||| */
                                /*|||||+++---------- : Setting of start trigger or capture trigger of channel 6 */
                                /*|||||  : 0 0 0 : Only software trigger start is valid (other trigger sources are unselected).
*/
                                /*|||||  : 0 0 1 : Valid edge of TI06 pin input signal, fSUB/2, fSUB/4, or INTRTC1 is used
as both the start trigger and capture trigger. */
                                /*|||||  : 0 1 0 : Both the edges of TI06 pin input signal, fSUB/2, fSUB/4, or INTRTC1 are
used as a start trigger and a capture trigger. */
                                /*|||||  : 1 0 0 : Interrupt signal of the master channel is used (when the channel is used
as a slave channel with the combination operation function). */
                                /*|||||  : Other than above : Setting prohibited */
                                /*||||| */
                                /*||||+---------- : Selection of slave/master of channel 6 */
                                /*||||  : 0 : Operates as slave channel with combination operation function. */
                                /*||||  : 1 : Operates as master channel with combination operation function. */
                                /*|||| */
                                /*|||+---------- : Selection of count clock (TCLK) of channel 0 */
                                /*|||  : 0 : Operation clock MCK specified by CKS04 bit */
                                /*|||  : 1 : Valid edge of input signal input from TI04 pin, fSUB/2, fSUB/4, or INTRTC1 */
                                /*|||             (the timer input used with channel 4 is selected by using TIS0 register). */
                                /*||| */
                                /*|++---------- : Be sure to set 00 */
                                /*| */
                                /*+---------- : Selection of operation clock (MCK) of channel 4 */
                                /*  : 0 : Operation clock CK00 set by TPS0 register */
                                /*  : 1 : Operation clock CK01 set by TPS0 register */
    /*      TDR06 = 2000;       /* set interval time to 100us(=0.05us * 2000) */
            TMMK06 = 1;                     /* disable interrupt */

    }

    /*---------------------------------------------------------------*/
    /* Module:          fn_InitLvi
                        */
    /* Description:     Setting of Low-voltage detectior                                    */
    /*      parameter: --
                                */
```

```
/*      return     : --
                                */
/*-----------------------------------------------------------------*/
void    fn_InitLvi(void)
{
        unsigned short      loop;       /* waiting counter */

        LVIMK = 1;                              /* disable LVI interrupt */

        LVIS =    0b00001001;                   /* Low-Voltage Detection Level Select Register */
                        /*||||++++--- : Detection level */
                        /*|||| : 0 0 0 0 : VLVI0 (4.22V) */
                        /*|||| : 0 0 0 1 : VLVI1 (4.07V) */
                        /*|||| : 0 0 1 0 : VLVI2 (3.92V) */
                        /*|||| : 0 0 1 1 : VLVI3 (3.76V) */
                        /*|||| : 0 1 0 0 : VLVI4 (3.61V) */
                        /*|||| : 0 1 0 1 : VLVI5 (3.45V) */
                        /*|||| : 0 1 1 0 : VLVI6 (3.30V) */
                        /*|||| : 0 1 1 1 : VLVI7 (3.15V) */
                        /*|||| : 1 0 0 0 : VLVI8 (2.99V) */
                        /*|||| : 1 0 0 1 : VLVI9 (2.84V) */
                        /*|||| : 1 0 1 0 : VLVI10 (2.68V) */
                        /*|||| : 1 0 1 1 : VLVI11 (2.53V) */
                        /*|||| : 1 1 0 0 : VLVI12 (2.38V) */
                        /*|||| : 1 1 0 1 : VLVI13 (2.22V) */
                        /*|||| : 1 1 1 0 : VLVI14 (2.07V) */
                        /*|||| : 1 1 1 1 : VLVI15 (1.91V) */
                        /*|||| */
                        /*++++------- : Be sure to set 0000 */

        LVIM =    0b10000000;                   /* Low-Voltage Detection Register */
                        /*|||||||+--- : LVIF Low-voltage detection flag */
                        /*||||||| : 0 : * LVISEL = 0: VDD >= VLVI, or when LVI operation is disabled */
                        /*|||||||            * LVISEL = 1: EXLVI >= VEXLVI, or when LVI operation is disabled */
                        /*||||||| : 1 : * LVISEL = 0: VDD < VLVI */
                        /*|||||||            * LVISEL = 1: EXLVI < VEXLVI */
                        /*||||||| */
                        /*||||||+---- : Low-voltage detection operation mode (interrupt/reset) selection(LVIMD) */
                        /*|||||| : 0 : * LVISEL = 0: Generates an internal interrupt signal */
```

```
                        /*||||||                    when VDD drops lower than VLVI (VDD < VLVI) */
                        /*||||||                    or when VDD becomes VLVI or higher (VDD >= VLVI).
*/
                        /*||||||        * LVISEL = 1: Generates an interrupt signal */
                        /*||||||                      when EXLVI drops lower than VEXLVI (EXLVI <
VEXLVI) */
                        /*||||||                    or when EXLVI becomes VEXLVI or higher (EXLVI >=
VEXLVI). */
                        /*|||||| : 1 : * LVISEL = 0: Generates an internal reset signal when VDD < VLVI */
                        /*||||||                    and releases the reset signal when VDD >= VLVI. */
                        /*||||||        * LVISEL = 1: Generates an internal reset signal when EXLVI < VEXLVI
*/
                        /*||||||                    and releases the reset signal when EXLVI >= VEXLVI.
*/
                        /*|||||| */
                        /*|||||+----- : Voltage detection selection(LVISEL) */
                        /*||||| : 0 : Detects level of supply voltage (VDD) */
                        /*||||| : 1 : Detects level of input voltage from external input pin (EXLVI) */
                        /*||||| */
                        /*|++++------ : Be sure to set 0000 */
                        /*| */
                        /*+---------- : Enables low-voltage detection operation */
                        /* : 0 : Disables operation */
                        /* : 1 : Enables operation */

        /* software to wait for the operation stabilization time (>210us) */
        fn_Wait500usBase(500/500);

        /* wait until VLVI≤VDD */
        while( LVIF ){
                NOP();
        }

        LVIIF = 0;                      /* clear LVI interrupt request flag */
    }


    /*-----------------------------------------------------------------*/
    /* Module:          fn_InitRtc
```

```
                    */
/* Description:     Setting of Real-time counter                                    */
/*       parameter: --
                                    */
/*       return     : --
                                    */
/*-----------------------------------------------------------*/
void    fn_InitRtc(void)
{
        RTCEN = 1;                          /* supplies operational real-time counter (RTC) input clock. */

        RTCC0 = 0b00001001;                 /* Real-Time Counter Control Register 0 */
                                    /*|||||+++--- : Constant-period interrupt (INTRTC) selection */
                                    /*||||| : 0 0 0 : Does not use constant-period interrupt function. */
                                    /*||||| : 0 0 1 : Once per 0.5 s */
                                    /*||||| : 0 1 0 : Once per 1 s */
                                    /*||||| : 0 1 1 : Once per 1 m */
                                    /*||||| : 1 0 0 : Once per 1 hour */
                                    /*||||| : 1 0 1 : Once per 1 day */
                                    /*||||| : 1 1 x : Once per 1 month */
                                    /*|||||    x = don't care */
                                    /*||||| */
                                    /*||||+------ : Selection of 12-/24-hour system */
                                    /*|||| : 0 : 12-hour system */
                                    /*|||| : 1 : 24-hour system */
                                    /*|||| */
                                    /*|||+------- : RTCCL pin output control */
                                    /*||| : 0 : Disables output of RTCCL pin (32 kHz). */
                                    /*||| : 1 : Enables output of RTCCL pin (32 kHz). */
                                    /*||| */
                                    /*||+-------- : RTC1HZ pin output control */
                                    /*|| : 0 : Disables output of RTC1HZ pin (1 Hz). */
                                    /*|| : 1 : Enables output of RTC1HZ pin (1 Hz). */
                                    /*|| */
                                    /*|+--------- : Be sure to set 0 */
                                    /*| */
                                    /*+---------- : Real-time counter operation control */
                                    /* : 0 : Stops counter operation. */
                                    /* : 1 : Starts counter operation. */
```

```
    RTCC2 =  0b10000011;                    /* Real-Time Counter Control Register 2 */
                                /*+llll+++--- : Constant-period interrupt (INTRTC) selection */
                                /* llll : 0 x x x : Interval interrupt is not generated. */
                                /* llll : 1 0 0 0 : 2^6/fSUB    (    1.953125 ms) */
                                /* llll : 1 0 0 1 : 2^7/fSUB    (    3.90625 ms) */
                                /* llll : 1 0 1 0 : 2^8/fSUB    (    7.8125 ms) */
                                /* llll : 1 0 1 1 : 2^9/fSUB    ( 15.625 ms) */
                                /* llll : 1 1 0 0 : 2^10/fSUB ( 31.25 ms) */
                                /* llll : 1 1 0 1 : 2^11/fSUB ( 62.5 ms) */
                                /* llll : 1 1 1 x : 2^12/fSUB ( 125 ms) */
                                /* llll    x = don't care */
                                /* llll */
                                /* ll++------ : Be sure to set 0 */
                                /* ll    */
                                /* l+-------- : Selection of RTCDIV pin output frequency */
                                /* l : 0 : RTCDIV pin outputs 512 Hz (1.95 ms). */
                                /* l : 1 : RTCDIV pin outputs 16.384 kHz (0.061 ms). */
                                /* l */
                                /* +--------- : RTCDIV pin output control */
                                /*    : 0 : Disables output of RTCDIV pin */
                                /*    : 1 : Enables output of RTCDIV pin */

    RTCMK = 1;                              /* disable RTC interrupt */
    RTCIF = 0;                              /* clear RTC interrupt request flag */

    RTCIMK = 1;                             /* disable RTCI interrupt */
    RTCIIF = 0;                             /* clear RTCI interrupt request flag */

}


/*---------------------------------------------------------------*/
/* Module:          fn_Wait500usBase
        */
/* Description:     Delays the program for (Time * 500us)                    */
/*      parameter: wait time(Time)
        */
/*      return     : --
```

```
                                        */
/*---------------------------------------------------------------*/
void    fn_Wait500usBase(unsigned short Time)
{
        TS0L.1 = 1;                             /* start TAU0 CH1 */
        TMIF01 = 0;

        for(; Time > 0; Time--){        /* wait for (parameter * 500)us */
                while(!TMIF01){
                        NOP();
                }
                TMIF01 = 0;
        }

        TT0L.1 = 1;                             /* stop TAU0 CH1 */
}


/*---------------------------------------------------------------*/
/* Module:          main
                                        */
/* Description:     Main process
        */
/*      parameter: --
                                        */
/*      return     : --
                                        */
/*---------------------------------------------------------------*/
void main(void)
{

        unsigned char       ucCount;            /* work counter */


        for(ucCount = 0; ucCount<4 ; ucCount++){
                ushColumnOff[ucCount] = 0xffff;                 /* Column Notouch data */
                ushColumnCurrent[ucCount] = 0xffff;     /* Column current data */
        }
```

```
	for(ucCount = 0; ucCount<2 ; ucCount++){
		ushRowOff[ucCount] = 0xffff;			/* Row Notouch data */
		ushRowCurrent[ucCount] = 0xffff;		/* Row current data */
	}

	ucTouchStatusColumnWork = 0;		/* touch column position work */
	ucTouchStatusRowWork = 0;		/* touch row position work */

	ucTouchStatusColumnLast = 0; /* last touch column position work */
	ucTouchStatusRowLast = 0;		/* last touch row position work */

	ucTouchStatusColumn = 0;		/* touch column position */
	ucTouchStatusRow = 0;				/* touch row position */

	EGP0 |= 0b10111110;					/* set INTP1-INTP5,INTP7 to Rising edge
*/
	EGN0 &= 0b01000001;					/* reset INTP1-INTP5,INTP7 to Falling
edge */



	/*====================================*/
	/*------------------------------------*/
	/*		Main Loop		*/
	/*------------------------------------*/
	/*====================================*/



	fn_Display(ucTouchStatusRow, ucTouchStatusColumn);		/* initial display */

	/*================================================================*/
	/* if system have nothing to do, go to standby for power-saving */
	/*================================================================*/


	while (1){

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

```
/* check P3.0(INTP1)  */
/*=======================*/

P3.0 = 0;                                   /* P3.0(INTP1) Latch Low */
PM3.0 = 0;                                  /* set P3.0 output to discharge */
fn_Wait500usBase(500/500);    /* wait discharge */

PIF1 = 0;                                   /* clear INTP1 request */
TS0L.6 = 1;                                 /* start TAU0 CH6 */
TMIF06 = 0;                                 /* clear TM6 interrupt request */
PM3.0 = 1;                                  /* set P3.0 input to start charge
*/
while(PIF1 == 0);                           /* wait charge end */
ushColumnCurrent[0] = TCR06;/* get timer count */
if(ushColumnOff[0] > ushColumnCurrent[0]){
        ushColumnOff[0] = ushColumnCurrent[0];  /* set touch pad off data */
}
TT0L.6 = 1;                                 /* stop TAU0 CH6 */

/*=======================*/
/* check P3.1(INTP2)  */
/*=======================*/

P3.1 = 0;                                   /* P3.1(INTP2) Latch Low */
PM3.1 = 0;                                  /* set P3.1 output to discharge */
fn_Wait500usBase(500/500);    /* wait discharge */

PIF2 = 0;                                   /* clear INTP2 request */
TS0L.6 = 1;                                 /* start TAU0 CH6 */
TMIF06 = 0;                                 /* clear TM6 interrupt request */
PM3.1 = 1;                                  /* set P3.1 input to start charge
*/
while(PIF2 == 0);                           /* wait charge end */
ushColumnCurrent[1] = TCR06;/* get timer count */
if(ushColumnOff[1] > ushColumnCurrent[1]){
        ushColumnOff[1] = ushColumnCurrent[1];  /* set touch pad off data */
}
TT0L.6 = 1;                                 /* stop TAU0 CH6 */
```

```
/*=======================*/
/* check P3.2(INTP5)  */
/*=======================*/

P3.2 = 0;                                       /* P3.2(INTP5) Latch Low */
PM3.2 = 0;                                              /* set P3.2 output to discharge */
fn_Wait500usBase(500/500);    /* wait discharge */

PIF5 = 0;                                       /* clear INTP5 request */
TS0L.6 = 1;                                             /* start TAU0 CH6 */
TMIF06 = 0;                                             /* clear TM6 interrupt request */
PM3.2 = 1;                                              /* set P3.2 input to start charge
*/
while(PIF5 == 0);                               /* wait charge end */
ushColumnCurrent[2] = TCR06;/* get timer count */
if(ushColumnOff[2] > ushColumnCurrent[2]){
        ushColumnOff[2] = ushColumnCurrent[2];  /* set touch pad off data */
}
TT0L.6 = 1;                                             /* stop TAU0 CH6 */


/*=======================*/
/* check P3.3(INTP3)  */
/*=======================*/

P3.3 = 0;                                       /* P3.3(INTP3) Latch Low */
PM3.3 = 0;                                              /* set P3.3 output to discharge */
fn_Wait500usBase(500/500);    /* wait discharge */

PIF3 = 0;                                       /* clear INTP3 request */
TS0L.6 = 1;                                             /* start TAU0 CH6 */
TMIF06 = 0;                                             /* clear TM6 interrupt request */
PM3.3 = 1;                                              /* set P3.3 input to start charge
*/
while(PIF3 == 0);                               /* wait charge end */
ushColumnCurrent[3] = TCR06;/* get timer count */
if(ushColumnOff[3] > ushColumnCurrent[3]){
        ushColumnOff[3] = ushColumnCurrent[3];  /* set touch pad off data */
```

```
    }
    TT0L.6 = 1;                                         /* stop TAU0 CH6 */


    /*=======================*/
    /* check P1.4(INTP4)  */
    /*=======================*/

    P1.4 = 0;                                       /* P1.4(INTP4) Latch Low */
    PM1.4 = 0;                                          /* set P1.4 output to discharge */
    fn_Wait500usBase(500/500);    /* wait discharge */

    PIF4 = 0;                                       /* clear INTP4 request */
    TS0L.6 = 1;                                         /* start TAU0 CH6 */
    TMIF06 = 0;                                         /* clear TM6 interrupt request */
    PM1.4 = 1;                                          /* set P1.4 input to start charge
*/
    while(PIF4 == 0);                           /* wait charge end */
    ushRowCurrent[0] = TCR06;     /* get timer count */
    if(ushRowOff[0] > ushRowCurrent[0]){
            ushRowOff[0] = ushRowCurrent[0];        /* set touch pad off data */
    }
    TT0L.6 = 1;                                         /* stop TAU0 CH6 */


    /*=======================*/
    /* check P1.5(INTP7)  */
    /*=======================*/

    P1.5 = 0;                                       /* P1.5(INTP7) Latch Low */
    PM1.5 = 0;                                          /* set P1.5 output to discharge */
    fn_Wait500usBase(500/500);    /* wait discharge */

    PIF7 = 0;                                       /* clear INTP7 request */
    TS0L.6 = 1;                                         /* start TAU0 CH6 */
    TMIF06 = 0;                                         /* clear TM6 interrupt request */
    PM1.5 = 1;                                          /* set P1.5 input to start charge
*/
    while(PIF7 == 0);                           /* wait charge end */
```

```
                ushRowCurrent[1] = TCR06;     /* get timer count */
                if(ushRowOff[1] > ushRowCurrent[1]){
                        ushRowOff[1] = ushRowCurrent[1];          /* set touch pad off data */
                }
                TT0L.6 = 1;                                                  /* stop TAU0 CH6 */



                /*================================*/
                /* check touch position                                  */
                /*================================*/

                ucTouchStatusColumnWork = 0;            /* touch column position work */
    /*          ucTouchStatusColumn = 0;                /* touch column position */

                for(ucCount = 0; ucCount < 4; ucCount++){
                        if((ushColumnCurrent[ucCount] - ushColumnOff[ucCount]) > 5){
                                                /* touch detect in column */
                                if(ucTouchStatusColumnWork != 0){
                                        ucTouchStatusColumnWork = 0xff;          /* multi position touch
detect */
                                        break;
                                }
                                else{
                                        ucTouchStatusColumnWork = (ucCount + 1);/* set touch
position */
                                }
                        }
                }

                ucTouchStatusRowWork = 0;    /* touch row position work */
                for(ucCount = 0; ucCount < 2; ucCount++){
                        if((ushRowCurrent[ucCount] - ushRowOff[ucCount]) > 5){
                                                /* touch detect in row */
                                if(ucTouchStatusRowWork != 0){
                                        ucTouchStatusRowWork = 0xff;/* multi position touch detect */
                                        break;
                                }
                                else{
                                        ucTouchStatusRowWork = (ucCount + 1);  /* set touch position */
```

```
                        }
                }
        }

                /* compare last detect position to remove noize */
        if((ucTouchStatusColumnLast == ucTouchStatusColumnWork) &&
                (ucTouchStatusRowLast == ucTouchStatusRowWork)
){
                ucTouchStatusColumn = ucTouchStatusColumnWork;
                ucTouchStatusRow = ucTouchStatusRowWork;
        }
                /* renew last status */
        ucTouchStatusColumnLast = ucTouchStatusColumnWork;
        ucTouchStatusRowLast = ucTouchStatusRowWork;

                /* notouch point adjust */
        if((ucTouchStatusColumn == 0x00)||(ucTouchStatusRow == 0x00)){
                ucTouchStatusColumn = 0x00;
                ucTouchStatusRow = 0x00;
        }

                /* multi touch point adjust */
        if((ucTouchStatusColumn == 0xff)||(ucTouchStatusRow == 0xff)){
                ucTouchStatusColumn = 0xff;
                ucTouchStatusRow = 0xff;
        }

        fn_Display(ucTouchStatusColumn, ucTouchStatusRow);          /* LCD display */

        /*=========================================*/
        /*    Change to Sub-HALT mode (about 15ms)          */
        /*=========================================*/

        DI();
        CKC = 0b01000000;               /* CPU/peripheral hardware clock is subclock */
        NOP();
        NOP();

        /* Confirming the CPU clock status */
```

```
            while(CLS != 1){
                    NOP();
            }

            MSTOP = 1;                                  /* X1 oscillator stopped */

            RTCE = 1;                           /* Starts counter operation. */
            RTCIIF = 0;                                 /* clear RTCI interrupt request flag */
            RTCIMK = 0;                                 /* enable RTCI interrupt */

            HALT();                                     /* Sub-HALT mode */
            NOP();
            NOP();

            RTCIMK = 1;                                 /* disable RTCI interrupt */

            RTCE = 0;                           /* Stops counter operation. */
            RTCIIF = 0;                                 /* clear RTCI interrupt request flag */

            MSTOP = 0;                                  /* X1 oscillator operating */
            while(OSTC.7 != 1){             /* wait X1 oscillation stabilization */
                    NOP();
            }
            CKC =   0b00010000;                 /* CPU/peripheral hardware clock is main clock */

            /* Confirming the CPU clock status */
            while((CLS != 0)||(MCS != 1)){
                    NOP();
            }
            EI();

        }
    }
```

● display.c

```
/*
 * Copyright (C) NEC Electronics Corporation 2006
 * NEC ELECTRONICS CONFIDENTIAL AND PROPRIETARY
 * All rights reserved by NEC Electronics Corporation.
 * This program must be used solely for the purpose for which
 * it was furnished by NEC Electronics Corporation. No part of this
 * program may be reproduced or disclosed to others, in any
 * form, without the prior written permission of NEC Electronics
 * Corporation. Use of copyright notice dose not evidence
 * publication of the program.
 */

/*------------------------------------------------------------------------*/
/* #pragma directive for CC78K0                                           */
/*------------------------------------------------------------------------*/
#pragma         SFR

/*------------------------------------------------------------------------*/
/* Include files                                                          */
/*------------------------------------------------------------------------*/
/*#include      "defines.h"*/
#include        <string.h>

/*------------------------------------------------------------------------*/
/* Function prototyps                                                     */
/*------------------------------------------------------------------------*/
                void    fn_DisplayAllClear(void);                       /* Display all clear */
extern  void    fn_Wait500usBase(unsigned short);   /* Delays the program (500us base) */

/*------------------------------------------------------------------------*/
/* Extern variables/constants                                             */
/*------------------------------------------------------------------------*/
/*------------------------------------------------------------------------*/
```

```
/* Local constants
                                                */
/*-------------------------------------------------------------------------*/
/* size of area for displaying */
#define CLCDSIZE_ALL       (unsigned char)(&SEG53 - &SEG4 + 1)     /* all area */
/*#define              CLCDSIZE_NUMBER              4                       /* number */
#define CLCDSIZE_NUMBER              5                    /* number */

/* Display starting position */
#define CLCDPOS_START              &SEG4     /* start of all area */
#define CLCDPOS_COLUMN2                      &SEG16   /* column 2 start */
#define CLCDPOS_COLUMN3                      &SEG28   /* column 3 start */
#define CLCDPOS_COLUMN4                      &SEG40   /* column 4 start */
#define CLCDPOS_END_SPACE        &SEG52   /* end space start */


/* Display data */
/*=========================================================================
   < LCD PANEL >

            SEG4     SEG5     SEG6     SEG7     SEG8       = = =       SEG49    SEG50    SEG51    SEG52
SEG53
          +------+------+------+------+------+-   = = =   -+------+------+------+------+------+
     COM0 |      |      |      |      |      |              |      |      |      |      |      |
          +------+------+------+------+------+-   = = =   -+------+------+------+------+------+
     COM1 |      |      |      |      |      |              |      |      |      |      |      |
          +------+------+------+------+------+-   = = =   -+------+------+------+------+------+
     COM2 |      |      |      |      |      |              |      |      |      |      |      |
          +------+------+------+------+------+-   = = =   -+------+------+------+------+------+
     COM3 |      |      |      |      |      |              |      |      |      |      |      |
          +------+------+------+------+------+-   = = =   -+------+------+------+------+------+
     COM4 |      |      |      |      |      |              |      |      |      |      |      |
          +------+------+------+------+------+-   = = =   -+------+------+------+------+------+
     COM5 |      |      |      |      |      |              |      |      |      |      |      |
          +------+------+------+------+------+-   = = =   -+------+------+------+------+------+
     COM6 |      |      |      |      |      |              |      |      |      |      |      |
          +------+------+------+------+------+-   = = =   -+------+------+------+------+------+
     COM7 |      |      |      |      |      |              |      |      |      |      |      |
          +------+------+------+------+------+-   = = =   -+------+------+------+------+------+
```

```
< example of the data setting >
    when you display "A" at area from SEG4 to SEG8

            +-----+-----+-----+-----+-----+
     bit0 |     |     |/////|     |     |      <- Blank area is 0.
            +-----+-----+-----+-----+-----+        Not blank area is 1.
     bit1 |     |/////|     |/////|     |
            +-----+-----+-----+-----+-----+
     bit2 |/////|     |     |     |/////|
            +-----+-----+-----+-----+-----+
     bit3 |/////|     |     |     |/////|
            +-----+-----+-----+-----+-----+
     bit4 |/////|/////|/////|/////|/////|
            +-----+-----+-----+-----+-----+
     bit5 |/////|     |     |     |/////|
            +-----+-----+-----+-----+-----+
     bit6 |/////|     |     |     |/////|
            +-----+-----+-----+-----+-----+
     bit7 |     |     |     |     |     |
            +-----+-----+-----+-----+-----+
               |     |     |     |     |
             0x7C  0x12  0x11  0x12  0x7C
               |     |     |     |     |
    SEG4 <---+     |     |     |     |
    SEG5 <---------+     |     |     |
    SEG6 <---------------+     |     |
    SEG7 <-------------------+     |
    SEG8 <-------------------------+
                  Write data to SFR

================================================================================*/
/*------------------------*/
/*    Number indication    */
/*------------------------*/
static const unsigned char aDispNumber[][CLCDSIZE_NUMBER] = {
/* COM 76543210 */
        {0b01111100                     /* '0' */
```

```
        ,0b10000010
        ,0b10000010
        ,0b01111100
        ,0b00000000}

/* COM 76543210 */
,       {0b00000100             /* '1' */
        ,0b00000100
        ,0b11111110
        ,0b00000000
        ,0b00000000}

/* COM 76543210 */
,       {0b11000100             /* '2' */
        ,0b10100010
        ,0b10010010
        ,0b10001100
        ,0b00000000}

/* COM 76543210 */
,       {0b01000100             /* '3' */
        ,0b10010010
        ,0b10010010
        ,0b01101100
        ,0b00000000}

/* COM 76543210 */
,       {0b01111000             /* '4' */
        ,0b01000100
        ,0b11111110
        ,0b01000000
        ,0b00000000}

/* COM 76543210 */
,       {0b01001110             /* '5' */
        ,0b10001010
        ,0b10001010
        ,0b01110010
        ,0b00000000}
```

```
/* COM 76543210 */
,       {0b01111100                     /* '6' */
        ,0b10010010
        ,0b10010010
        ,0b01100000
        ,0b00000000}

/* COM 76543210 */
,       {0b00000010                     /* '7' */
        ,0b11100010
        ,0b00011010
        ,0b00000110
        ,0b00000000}

/* COM 76543210 */
,       {0b01101100                     /* '8' */
        ,0b10010010
        ,0b10010010
        ,0b01101100
        ,0b00000000}

/* COM 76543210 */
,       {0b00001100                     /* '9' */
        ,0b10010010
        ,0b01010010
        ,0b00111100
        ,0b00000000}
};



/*-------------------------*/
/*    TouchPosition frame      */
/*-------------------------*/
static const unsigned char aDispTouchPositionFrame[] = {
/* COM 76543210 */
          0b00000000
        ,0b01111111                     /* ------- */
```

```
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01111111         /* ------- */
,0b00000000
,0b01111111         /* ------- */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01111111         /* ------- */
,0b00000000
,0b01111111         /* ------- */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01111111         /* ------- */
,0b00000000
,0b01111111         /* ------- */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
,0b01001001         /* |   |   | */
```

```
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01111111                 /* ------- */
        ,0b00000000
        ,0b00000000
};

static const unsigned char aDispTouchPositionOffFrame[] = {
/* COM 76543210 */
         0b00000000
        ,0b01111111                 /* ------- */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01001001                 /* |   |   | */
        ,0b01111111                 /* ------- */
};

static const unsigned char aDispTouchPositionUpperOnFrame[] = {
/* COM 76543210 */
         0b00000000
        ,0b01111111                 /* ------- */
        ,0b01001111                 /* |   |||| */
        ,0b01001111                 /* |   |||| */
        ,0b01001111                 /* |   |||| */
        ,0b01001111                 /* |   |||| */
        ,0b01001111                 /* |   |||| */
        ,0b01001111                 /* |   |||| */
        ,0b01001111                 /* |   |||| */
        ,0b01001111                 /* |   |||| */
```

```
        ,0b01001111                     /* |    |||| */
        ,0b01111111                     /* ------- */
};

static const unsigned char aDispTouchPositionLowerOnFrame[] = {
/* COM 76543210 */
         0b00000000
        ,0b01111111                     /* ------- */
        ,0b01111001                     /* ||||    | */
        ,0b01111001                     /* ||||    | */
        ,0b01111001                     /* ||||    | */
        ,0b01111001                     /* ||||    | */
        ,0b01111001                     /* ||||    | */
        ,0b01111001                     /* ||||    | */
        ,0b01111001                     /* ||||    | */
        ,0b01111001                     /* ||||    | */
        ,0b01111001                     /* ||||    | */
        ,0b01111111                     /* ------- */
};

static const unsigned char aDispTouchPositionMultiFrame[] = {
/* COM 76543210 */
         0b00000000
        ,0b01111111                     /* ------- */
        ,0b01111111                     /* ||||||| */
        ,0b01111111                     /* ||||||| */
        ,0b01111111                     /* ||||||| */
        ,0b01111111                     /* ||||||| */
        ,0b01111111                     /* ||||||| */
        ,0b01111111                     /* ||||||| */
        ,0b01111111                     /* ||||||| */
        ,0b01111111                     /* ||||||| */
        ,0b01111111                     /* ||||||| */
        ,0b01111111                     /* ------- */
};
```

```
static const unsigned char aDispClear[] = {
/* COM 76543210 */
             0b00000000                  /* ' ' */
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000                  /* ' ' */
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000                  /* ' ' */
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000                  /* ' ' */
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000                  /* ' ' */
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000                  /* ' ' */
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000                  /* ' ' */
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000
            ,0b00000000                  /* ' ' */
            ,0b00000000
```

```
        ,0b00000000
        ,0b00000000
        ,0b00000000
        ,0b00000000                 /* ' ' */
        ,0b00000000
        ,0b00000000
        ,0b00000000
        ,0b00000000
        ,0b00000000                 /* ' ' */
        ,0b00000000
        ,0b00000000
        ,0b00000000
        ,0b00000000
};

/*---------------------------------------------------------------------*/
/* Global variables                                                    */
/*---------------------------------------------------------------------*/

/*---------------------------------------------------------------------*/
/* Local variables                                                     */
/*---------------------------------------------------------------------*/

/*---------------------------------------------------------------------*/
/* Code                                                                */
/*---------------------------------------------------------------------*/
/*=====================================================================*/
/*                                                                     */
/*                          Common function                            */
/*                                                                     */
/*=====================================================================*/
/*---------------------------------------------------------------*/
/* Module:          fn_DisplayInit                               */
/* Description:     Initialization of Display module             */
/*        parameter: --
```

```
                                */
/*      return     : --
                                */
/*-----------------------------------------------------------------*/
void    fn_DisplayInit(void)
{
        /*---------------------------------------*/
        /* Initialization of LCD controler/driver */
        /*---------------------------------------*/
        LCDMD = 0b00010000;                     /* LCD Mode Register */
                                /*++||++++--- : Be sure to set 000000 */
                                /*   || */
                                /*   ++------- : LCD drive voltage generator selection */
                                /*     : 0 0 : External resistance division method */
                                /*     : 0 1 : Internal voltage boosting method */
                                /*     : 1 0 : Capacitance split method */
                                /*     : 1 1 : Setting prohibited */
        SEGEN = 0b00011111;                     /* Segment Enable Register */
                                /*|||||||+--- : Control segment signal output from pins SEG8-SEG11 */
                                /*||||||+---- : Control segment signal output from pins SEG12-SEG15 */
                                /*|||||+----- : Control segment signal output from pins SEG16-SEG19 */
                                /*||||+------ : Control segment signal output from pins SEG20-SEG23 */
                                /*|||+------- : Control segment signal output from pins SEG24-SEG26 */
                                /*||| : 0 : segment signal output disable */
                                /*||| : 1 : segment signal output enable */
                                /*||| */
                                /*+++-------- : Be sure to set 000 */
        PFALL =   0b01111111;                   /* Port Function Register ALL */
                                /*|||||||+--- : Pins P50-P53 port/segment output specification */
                                /*||||||+---- : Pins P54-P57 port/segment output specification */
                                /*|||||+----- : Pins P90-P93 port/segment output specification */
                                /*||||+------ : Pins P94-P97 port/segment output specification */
                                /*|||+------- : Pins P100-P102 port/segment output specification */
                                /*||+-------- : Pins P140-P143 port/segment output specification */
                                /*|+--------- : Pins P144-P147 port/segment output specification */
                                /*| : 0 : Used the pins as port (other than segment output) */
                                /*| : 1 : Used the pins as segment output */
                                /*| */
                                /*+---------- : Be sure to set 0 */
```

```
ISC =   0b00000000;                 /* Input Switch Control Register */
                    /*|||||||+--- : Switching external interrupt (INTP0) input */
                    /*|||||||  : 0 : Uses the input signal of the INTP0 pin as an external interrupt (normal
operation). */
                    /*|||||||  : 1 : Uses the input signal of the RXD3 pin as an external interrupt */
                    /*|||||||      (to measure the pulse widths of the sync break field and sync field). */
                    /*||||||| */
                    /*||||||+---- : ISC1 Switching channel 7 input of timer array unit TAUS */
                    /*||||||  : 0 : Uses the input signal of the TI07 pin as a timer input (normal operation). */
                    /*||||||  : 1 : Input signal of RXD3 pin is used as timer input (wakeup signal detection). */
                    /*|||||| */
                    /*|||||+----- : RxD3/SEG53/P50 pin schmitt-triggered buffer control */
                    /*|||||  : 0 : Disables input */
                    /*|||||  : 1 : Enables input */
                    /*||||| */
                    /*||||+------ : TI02/SEG51/P52 pin schmitt-triggered buffer control */
                    /*||||  : 0 : Disables input */
                    /*||||  : 1 : Enables input */
                    /*|||| */
                    /*|||+------- : TI04/SEG50/P53 pin schmitt-triggered buffer control */
                    /*|||  : 0 : Disables input */
                    /*|||  : 1 : Enables input */
                    /*||| */
                    /*+++-------- : Be sure to set 000 */

fn_DisplayAllClear();           /* clear all LCD String area */

LCDC0 =  0b00000011;                /* LCD Clock Control Register */
                    /*|||||+++--- : LCD clock (LCDCL) selection */
                    /*|||||  : 0 0 0 : fLCD/2^4 */
                    /*|||||  : 0 0 1 : fLCD/2^5 */
                    /*|||||  : 0 1 0 : fLCD/2^6 */
                    /*|||||  : 0 1 1 : fLCD/2^7 */
                    /*|||||  : 1 0 0 : fLCD/2^8 */
                    /*|||||  : 1 0 1 : fLCD/2^9 */
                    /*|||||  : Other than above : Setting prohibited */
                    /*||||| */
                    /*++||+------ : Be sure to set 000 */
                    /*   || */
```

```
                    /*   ++------- : LCD source clock (fLCD) selection */
                    /* : 0 0 : fSUB */
                    /* : 0 1 : fCLK/2^6 */
                    /* : 1 0 : fCLK/2^7 */
                    /* : 1 1 : fCLK/2^8 */

VLCD = 0x0A;                        /* set LCD boost level to 5V */
/* wait for the reference voltage setup time (2ms(min.)) */
fn_Wait500usBase(2000/500);

LCDM =   0b00100111;                /* LCD Display Mode Register */
                    /*|||||+++--- : LCD controller/driver display mode selection */
                    /*|||||   *When the external resistance division method is used */
                    /*|||||   : 0 0 0 : Four-time-slice mode & 1/3 bias method */
                    /*|||||   : 0 0 1 : Three-time-slice mode & 1/3 bias method */
                    /*|||||   : 0 1 0 : Two-time-slice mode & 1/2 bias method */
                    /*|||||   : 0 1 1 : Three-time-slice mode & 1/2 bias method */
                    /*|||||   : 1 0 0 : Static */
                    /*|||||   : 1 1 1 : Eight-time-slice mode & 1/4 bias method */
                    /*||||| */
                    /*|||||   *When the internal voltage boosting method is used */
                    /*|||||   : 0 0 0 : Four-time-slice mode & 1/3 bias method */
                    /*|||||   : 0 0 1 : Three-time-slice mode & 1/3 bias method */
                    /*|||||   : 0 1 0 : Four-time-slice mode & 1/3 bias method */
                    /*|||||   : 0 1 1 : Four-time-slice mode & 1/3 bias method */
                    /*|||||   : 1 0 0 : Setting prohibited */
                    /*|||||   : 1 1 1 : Eight-time-slice mode & 1/4 bias method */
                    /*||||| */
                    /*|||||   *When the capacitor split method is used */
                    /*|||||   : 0 0 0 : Four-time-slice mode & 1/3 bias method */
                    /*|||||   : 0 0 1 : Three-time-slice mode & 1/3 bias method */
                    /*|||||   : 0 1 0 : Four-time-slice mode & 1/3 bias method */
                    /*|||||   : 0 1 1 : Four-time-slice mode & 1/3 bias method */
                    /*|||||   : 1 0 0 : Setting prohibited */
                    /*|||||   : 1 1 1 : Four-time-slice mode & 1/3 bias method */
                    /*||||| */
                    /*|||||   : Other than above : Setting prohibited
                    /*||||| */
                    /*|||++------ : LCD display data area control */
```

```
                        /*lll : 0 0 : Display the data of an A pattern area */
                        /*lll           (lower 4 bits of LCD display data memory) */
                        /*lll : 0 1 : Display the data of an A pattern area */
                        /*lll           (higher 4 bits of LCD display data memory) */
                        /*lll : 1 0 : Display the data of an A pattern area and the B pattern area in turn. */
                        /*lll           (The on and off light indication which synchronized */
                        /*lll            in a constant-period interrupt timing of RTC) */
                        /*lll : 1 1 : Display the data of an A pattern area and the B pattern area in turn. */
                        /*lll           (The on and off light indication which synchronized */
                        /*lll            in a constant-period interrupt timing of RTC) */
                        /*lll */
                        /*ll+-------- : Voltage boost circuit and capacitor split circuit operation enable/disable */
                        /*ll : 0 : Stops voltage boost circuit and capacitor split circuit operation */
                        /*ll : 1 : Enables voltage boost circuit and capacitor split circuit operation */
                        /*ll */
                        /*++--------- : LCD display enable/disable */
                        /* : 0 0 : Output ground level to segment/common pin */
                        /* : 0 1 : Display off (all segment outputs are deselected.) */
                        /* : 1 0 : Output ground level to segment/common pin */
                        /* : 1 1 : Display on */

        /* software to wait for the operation stabilization time (over 500ms) */
        fn_Wait500usBase(500000/500);

        SCOC = 1;                       /* output deselect level to SEG and LCD waveform to COM */
        LCDON = 1;                      /* display on */

        /*-----------------------------------------*/
        /*      Initialization of variables        */
        /*-----------------------------------------*/
/*      bDispUpdate = 1;            /* initialize display update request flag */
/*      bBlinkOn = 0;                   /* initialize blink on flag */
/*      ushOpeningMessage = 0;      /* initialize opening message counter */
}

/*-------------------------------------------------------------*/
/* Module:          fn_LcdWrite
                                */
/* Description:     Write data to LCD RAM
```

```
        */
/*      parameter: position in whitch display begins                    */
/*                              address of display data
        */
/*                              size of display data
                    */
/*      return  : --
                              */
/*---------------------------------------------------------------*/
static void fn_LcdWrite(unsigned char *Position, unsigned char *DataAddr, unsigned char DataSize)
{
        memcpy(Position, DataAddr, DataSize);
}

/*---------------------------------------------------------------*/
/* Module:          fn_DisplayAllClear
        */
/* Description:     LCD RAM all clear
        */
/*      parameter: --
                              */
/*      return  : --
                              */
/*---------------------------------------------------------------*/
void fn_DisplayAllClear(void)
{
        fn_LcdWrite(CLCDPOS_START, aDispClear, CLCDSIZE_ALL);
}


/*---------------------------------------------------------------*/
/* Module:          fn_Display
                    */
/* Description:     Display touch position to LCD                  */
/*      parameter: touch column position, touch row position*/
/*      return  : --
                              */
/*---------------------------------------------------------------*/
void fn_Display(unsigned char ucColumn, unsigned char ucRow)
```

```
    {

        /*--------------------------*/
        /*          column 1 display                */
        /*--------------------------*/

        if(ucColumn == 1){
            if(ucRow == 1){     /* column 1 ,row 1 position on display */
                fn_LcdWrite(CLCDPOS_START,          aDispTouchPositionUpperOnFrame,
sizeof(aDispTouchPositionUpperOnFrame));
            }
            else{                       /* column 1 ,row 2 position on display */
                fn_LcdWrite(CLCDPOS_START,          aDispTouchPositionLowerOnFrame,
sizeof(aDispTouchPositionLowerOnFrame));
            }
        }
        else{
            if(ucColumn == 0xff){/* column 1 multiposition on display */
                fn_LcdWrite(CLCDPOS_START,              aDispTouchPositionMultiFrame,
sizeof(aDispTouchPositionMultiFrame));
            }
            else{                                       /* column 1 position off display */
                fn_LcdWrite(CLCDPOS_START,              aDispTouchPositionOffFrame,
sizeof(aDispTouchPositionOffFrame));
            }
        }



        /*--------------------------*/
        /*          column 2 display                */
        /*--------------------------*/

        if(ucColumn == 2){
            if(ucRow == 1){     /* column 2 ,row 1 position on display */
                fn_LcdWrite(CLCDPOS_COLUMN2,        aDispTouchPositionUpperOnFrame,
sizeof(aDispTouchPositionUpperOnFrame));
            }
            else{                       /* column 2 ,row 2 position on display */
                fn_LcdWrite(CLCDPOS_COLUMN2,        aDispTouchPositionLowerOnFrame,
```

```
sizeof(aDispTouchPositionLowerOnFrame));
                }
        }
        else{                                   /* column 2 multiposition on display */
                if(ucColumn == 0xff){
                        fn_LcdWrite(CLCDPOS_COLUMN2,            aDispTouchPositionMultiFrame,
sizeof(aDispTouchPositionMultiFrame));
                }
                else{                           /* column 2 position off display */
                        fn_LcdWrite(CLCDPOS_COLUMN2,            aDispTouchPositionOffFrame,
sizeof(aDispTouchPositionOffFrame));
                }
        }

        /*-------------------------*/
        /*      column 3 display           */
        /*-------------------------*/

        if(ucColumn == 3){
                if(ucRow == 1){         /* column 3 ,row 1 position on display */
                        fn_LcdWrite(CLCDPOS_COLUMN3,            aDispTouchPositionUpperOnFrame,
sizeof(aDispTouchPositionUpperOnFrame));
                }
                else{                           /* column 3 ,row 2 position on display */
                        fn_LcdWrite(CLCDPOS_COLUMN3,            aDispTouchPositionLowerOnFrame,
sizeof(aDispTouchPositionLowerOnFrame));
                }
        }
        else{                                   /* column 3 multiposition on display */
                if(ucColumn == 0xff){
                        fn_LcdWrite(CLCDPOS_COLUMN3,            aDispTouchPositionMultiFrame,
sizeof(aDispTouchPositionMultiFrame));
                }
                else{                           /* column 3 position off display */
                        fn_LcdWrite(CLCDPOS_COLUMN3,            aDispTouchPositionOffFrame,
sizeof(aDispTouchPositionOffFrame));
                }
        }
```

```
        /*---------------------------*/
        /*      column 4 display     */
        /*---------------------------*/

        if(ucColumn == 4){
                if(ucRow == 1){     /* column 4 ,row 1 position on display */
                        fn_LcdWrite(CLCDPOS_COLUMN4,        aDispTouchPositionUpperOnFrame,
sizeof(aDispTouchPositionUpperOnFrame));
                }
                else{                       /* column 4 ,row 2 position on display */
                        fn_LcdWrite(CLCDPOS_COLUMN4,        aDispTouchPositionLowerOnFrame,
sizeof(aDispTouchPositionLowerOnFrame));
                }
        }
        else{                               /* column 4 multiposition on display */
                if(ucColumn == 0xff){
                        fn_LcdWrite(CLCDPOS_COLUMN4,          aDispTouchPositionMultiFrame,
sizeof(aDispTouchPositionMultiFrame));
                }
                else{                       /* column 4 position off display */
                        fn_LcdWrite(CLCDPOS_COLUMN4,          aDispTouchPositionOffFrame,
sizeof(aDispTouchPositionOffFrame));
                }
        }

        /*---------------------------*/
        /*      remain area display  */
        /*---------------------------*/

        fn_LcdWrite(CLCDPOS_END_SPACE, aDispClear, 2);          /* remain all space */

    }
```

APPENDIX B REVISION HISTORY

Edition	Date Published	Page	Revision
1st edition	March 2010	–	–

Published by: NEC Electronics Corporation (http://www.necel.com/)
Contact: http://www.necel.com/support/