

[Notes]

R20TS0881EJ0100

Rev.1.00

Integrated Development Environment for Capacitive Touch

Dec. 16, 2022

Workbench6

Notes on R_Set_Cap_Touch_Initial_Tuning

Outline

When using the product in the title, note the following point.

- Notes on using the function R_Set_Cap_Touch_Initial_Tuning

1. Notes on Using the Function R_Set_Cap_Touch_Initial_Tuning

1.1 Applicable Products

Integrated Development Environment for Capacitive Touch Workbench6 Ver 1.04.00.00 and later

1.2 Applicable Devices

RX Family: RX113, RX130, RX230, RX231

1.3 Details

The following problems might occur due to an error in the processing after the condition ending the function "initial_offset_tuning", which is called in "R_Set_Cap_Touch_Initial_Tuning".

(1) The CTSUSO0.CTSUSO[9:0] bit is set to a value offset by 1. Subsequently, either of the following occurs:

(1-a) The status of the touch button is determined as continually ON.

(1-b) The touch button becomes inactive for a while or the sensitivity specified during sensitivity adjustment is changed.

In the case of (1-b), if the drift process is enabled, the specified sensitivity is resumed after some time.

(2) Adjustment of "R_Set_Cap_Touch_Initial_Tuning" takes long.

1.4 Conditions

The problem in 1.3 (1-a) might occur when all the following are met:

- The self-capacitance method is used.
- For the board constant (parasitic capacitance), "R_Set_Cap_Touch_Initial_Tuning" sets the CTSUSO0.CTSUSO[9:0] bit to "0x010*2ⁿ-1" (n = 1 to 31).
- The touch threshold and the CTSUSO0.CTSUSO[9:0] bit adjustment result meet the values in the following table.

Touch threshold	CTSUSO0.CTSUSO[9:0] bit adjustment result
250 to 562 or less	0x01F, 0x21F, 0x05F, 0x25F, 0x09F, 0x29F, 0x0DF, 0x2DF, 0x11F, 0x31F, 0x15F, 0x35F, 0x1BF, 0x3BF, 0x1DF, 0x3DF
796 or less	0x03F, 0x0BF, 0x13F, 0x1BF, 0x23F, 0x2BF, 0x33F, 0x3BF
1125 or less	0x07F, 0x17F, 0x27F, 0x37F
1593 or less	0x0FF, 0x2FF
2251 or less	0x1FF

1.5 Workaround

"initial_offset_tuning" adjusts offset values by hardware by increasing or decreasing the value of the CTSUSO0.CTSUSO[9:0] bit by 1 while referring to the measurement value so that the touch detection reference value becomes the target value (from 15510 to 15260).

Whether to continue offset adjustment is determined every ten calls. As an end condition, if the measurement is out of the range of target values and integration of the increased and decreased amount of CTSUSO0.CTSUSO[9:0] bit value is within ± 2 when vibration is converged, offset adjustment is deemed completed.

For measurement after completion of offset adjustment, the CTSUSO0.CTSUSO[9:0] bit is fixed to the value upon completion of the adjustment, and the measurement value with this setting will be the touch reference value. In this end condition, adjustment is deemed completed while the CTSUSO0.CTSUSO[9:0] bit is set to the next adjustment value. Therefore, a problem might occur in the next measurement due to a difference between the touch reference value and the measurement value.

To avoid such problems, modify "initial_offset_tuning" as follows.

Before modification: "initial_offset_tuning" of "r_ctsu.c"

```

uint8_t initial_offset_tuning( uint8_t method, uint8_t number )
{
  (Omitted)
  for (loop = 0; loop < number; loop++)
  {
    (Omitted)
    if (beas_val < (sensor_raw[loop] - TUNING_UPPER_LIMIT)) /* Current over check */
    {
      if (0x03FF != *(g_touch_tuning_info[method].ctsuso + loop)) /* CTSUS0 limit check */
      {
        *(g_dtc_write_data[method] + st) = (*(g_dtc_write_data[method] + st) & 0xFC00) +
        (*(g_touch_tuning_info[method].ctsuso + loop) + 1);
        *(g_touch_tuning_info[method].result + loop) = 0;
        *(g_current_sign_pt[method] + loop) = *(g_current_sign_pt[method] + loop) + 1; /* Plus */
      }
      else
      {
        *(g_touch_tuning_info[method].result + loop) = 1; /* Tuning finish flag set */
      }
    }
    else if (beas_val > (sensor_raw[loop] + TUNING_LOWER_LIMIT)) /* Current down check */
    {
      if (0x0000 != *(g_touch_tuning_info[method].ctsuso + loop)) /* CTSUS0 limit check */
      {
        *(g_dtc_write_data[method] + st) = (*(g_dtc_write_data[method] + st) & 0xFC00) +
        (*(g_touch_tuning_info[method].ctsuso + loop) - 1);
        *(g_touch_tuning_info[method].result + loop) = 0;
        *(g_current_sign_pt[method] + loop) = *(g_current_sign_pt[method] + loop) - 1;
      }
      else
      {
        *(g_touch_tuning_info[method].result + loop) = 1;
      }
    }
    else
    {
      *(g_touch_tuning_info[method].result + loop) = 1;
    }
  }
  if (10 == g_current_offset_count[method])
  {
    if ((OFFSET_CNT_PLUS >= *(g_current_sign_pt[method] + loop)) && (OFFSET_CNT_MINUS <=
    (*(g_current_sign_pt[method] + loop))))
    {
      *(g_touch_tuning_info[method].result + loop) = 1;
    }
    else
    {
      g_current_offset_count[method] = 0;
      *(g_current_sign_pt[method] + loop) = 0;
    }
  }
  if (0 == g_key_info[method].mode)
  {
    pt = pt + 1;
  }
  else
  {
    pt = pt + 3;
  }
  st = st + 3;
}
g_ctsu_status[method].flag.data_update = 0;
(Omitted)
} /* End of function initial_offset_tuning() */

```

Loop for the number of valid channels

Update CTSUSO0.CTSUSO[9:0] bit

Integration of increased/decreased amount of CTSUSO0.CTSUSO[9:0] bit

Integration of increased/decreased amount of CTSUSO0.CTSUSO[9:0] bit

Continuation determination counter (every 10 calls)

End condition that might cause a problem Because the continuation determination counter and integration of increased/decreased amount of CTSUSO0.CTSUSO bit are not cleared, the counter and integrated value might overflow.

The CTSUSO bit is updated. Adjustment is determined to be completed while the measurement value is unchecked.

During the loops (the number of valid channels), the continuation determination counter is cleared by the result of the channel being determined. As a result, determination of the next channel is suspended for the next 10 calls, thus extending the time for initialization.

After modification: "initial_offset_tuning" of "r_ctsu.c". Add the processing shown in red.

```

uint8_t initial_offset_tuning( uint8_t method, uint8_t number )
{
    (Omitted)
    for (loop = 0; loop < number; loop++)
    {
        (Omitted)
        if (beas_val < (sensor_raw[loop] - TUNING_UPPER_LIMIT))           /* Current over check */
        {
            if (0x03FF != *(g_touch_tuning_info[method].ctsuso + loop)) /* CTSUS0 limit check */
            {
                *(g_dtc_write_data[method] + st) = (*(g_dtc_write_data[method] + st) & 0xFC00) +
                (*(g_touch_tuning_info[method].ctsuso + loop) + 1);
                *(g_touch_tuning_info[method].result + loop) = 0;
                *(g_current_sign_pt[method] + loop) = *(g_current_sign_pt[method] + loop) + 1; /* Plus */
            }
            else
            {
                *(g_touch_tuning_info[method].result + loop) = 1; /* Tuning finish flag set */
            }
        }
        else if (beas_val > (sensor_raw[loop] + TUNING_LOWER_LIMIT))     /* Current down check */
        {
            if (0x0000 != *(g_touch_tuning_info[method].ctsuso + loop)) /* CTSUS0 limit check */
            {
                *(g_dtc_write_data[method] + st) = (*(g_dtc_write_data[method] + st) & 0xFC00) +
                (*(g_touch_tuning_info[method].ctsuso + loop) - 1);
                *(g_touch_tuning_info[method].result + loop) = 0;
                *(g_current_sign_pt[method] + loop) = *(g_current_sign_pt[method] + loop) - 1;
            }
            else
            {
                *(g_touch_tuning_info[method].result + loop) = 1;
            }
        }
        else
        {
            *(g_touch_tuning_info[method].result + loop) = 1;
        }
    }

    if (10 == g_current_offset_count[method])
    {
        if ((OFFSET_CNT_PLUS >= *(g_current_sign_pt[method] + loop)) && (OFFSET_CNT_MINUS <=
        *(g_current_sign_pt[method] + loop)))
        {
            *(g_touch_tuning_info[method].result + loop) = 1;
            *(g_dtc_write_data[method] + st) = (*(g_dtc_write_data[method] + st) & 0xFC00) +
            *(g_touch_tuning_info[method].ctsuso + loop);
        }
        else
        {
            // g_current_offset_count[method] = 0;
            *(g_current_sign_pt[method] + loop) = 0;
        }
    }

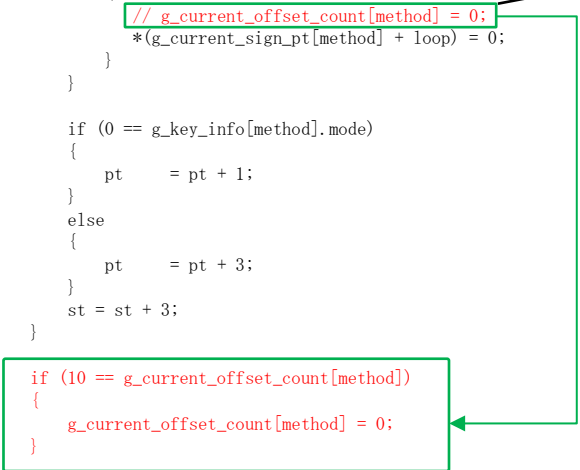
    if (0 == g_key_info[method].mode)
    {
        pt = pt + 1;
    }
    else
    {
        pt = pt + 3;
    }
    st = st + 3;
}

if (10 == g_current_offset_count[method])
{
    g_current_offset_count[method] = 0;
}

g_ctsu_status[method].flag.data_update = 0;
(Omitted)
} /* End of function initial_offset_tuning() */

```

Delete this



1.6 Schedule for Fixing the Problem

Since November 2018, Workbench6 and QE for Capacitive Touch have been released in parallel as capacitive touch development assistance tools. Note that while QE for Capacitive Touch will continue to be updated, Workbench6 will no longer be updated.

Problems in Workbench6 will not be fixed. If you continue to use Workbench6, apply the workaround.

It is recommended to use QE for Capacitive Touch for new projects. The above problems do not occur in QE for Capacitive Touch.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec.16.22	-	First edition issued

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.

The past news contents have been based on information at the time of publication. Now changed or invalid information may be included.

The URLs in the Tool News also may be subject to change or become invalid without prior notice.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/