AN204096

Built-In CR Clock Calibration for the Traveo™ Family

Author: Hiroo Mizuno

Associated Part Family: Traveo Family S6J3110/3120/3200/3310/3320/3330/3340/3350/3360/3370/3400/3510 Series

Related Documents: For a complete list, click here.

AN204096 explains how to calibrate the built-in CR clock of the Traveo™ family S6J3110/S6J3120/S6J3200/ S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 series using the CR calibration function.

Contents

1 Introduction

The Traveo family S6J3110/S6J3120/S6J3200/S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 series microcontrollers have a built-in high-speed CR oscillator or a built-in slow-speed CR oscillator. The details of these products are described in Table 1. The built-in CR oscillators can correct the frequency of the CR clock by configuring trimming. A trimming value can be determined from the calculation of a count value for correcting the frequency of the CR clock.

Table 1. CR Oscillator Type

Product Model	Built-in CR Oscillator Correction Type
S6J3110/S6J3120/S6J3200/S6J3300/S6J3350	High-speed CR oscillator
S6J3360/S6J3370/S6J3400/S6J3510	High-speed CR oscillator and slow-speed CR oscillator

This application note describes how to calibrate the built-in CR clock using the CR calibration function.

2 CR Calibration Function

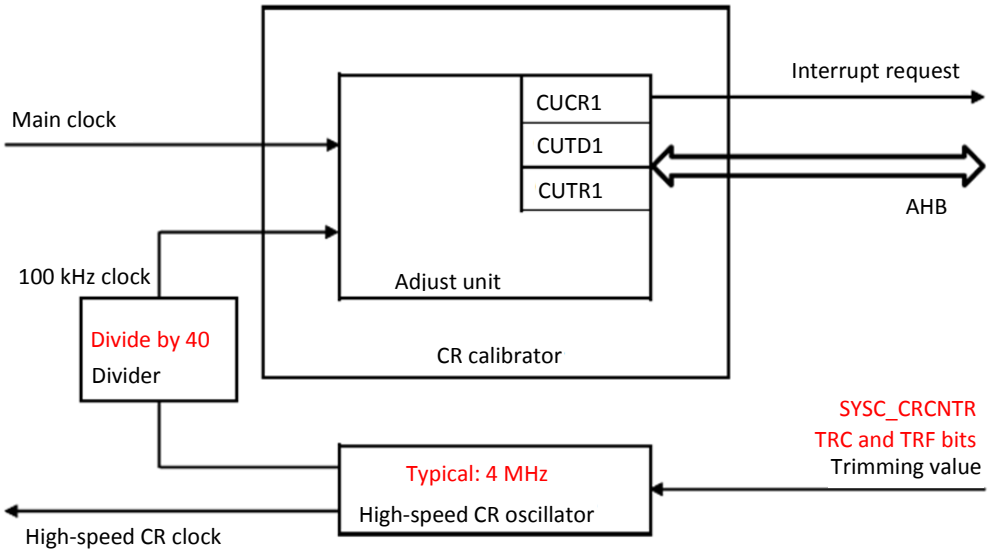
2.1 Configuration

Figure 1 shows the CR calibration block in the S6J3110/S6J3120/S6J3200/S6J3300/S6J3350/ S6J3360/S6J3370/ S6J3400/S6J3510 series. The built-in high-speed CR oscillator frequency is typically 4 MHz. The built-in slow-speed CR oscillator frequency is typically 100 kHz. These are adjusted by the trimming value that is stored in the CR clock control register (SYSC_CRCNTR) or the slow-CR clock control register (SYSC_SCRCNTR) in the clock system. SYSC_CRCNTR and SYSC_SCRCNTR have TRC and TRF bits respectively. The divided built-in high-speed CR clock is input to the adjust unit of the CR calibrator. The input clock is compared with the main clock. The CR calibrator can generate an interrupt request after comparing with the input clock and the main clock.

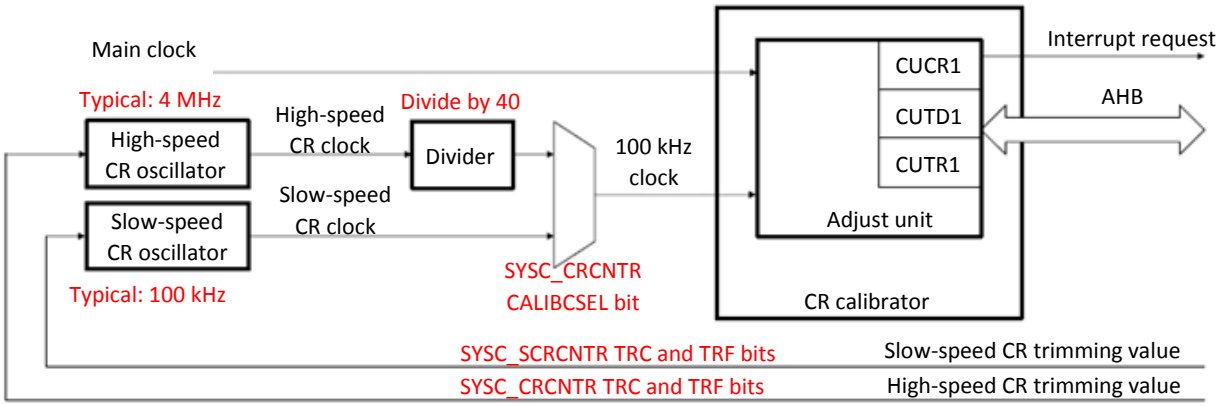
The S6J3360/S6J3370/S6J3400/S6J3510 series can select the input clock as the high-speed CR clock or the slow-speed CR clock with the CALIBCSEL bit of SYSC_CRCNTR.

Figure 1. CR Calibration Block Diagram:

S6J3110/S6J3120/S6J3200/S6J3300/S6J3350 Series



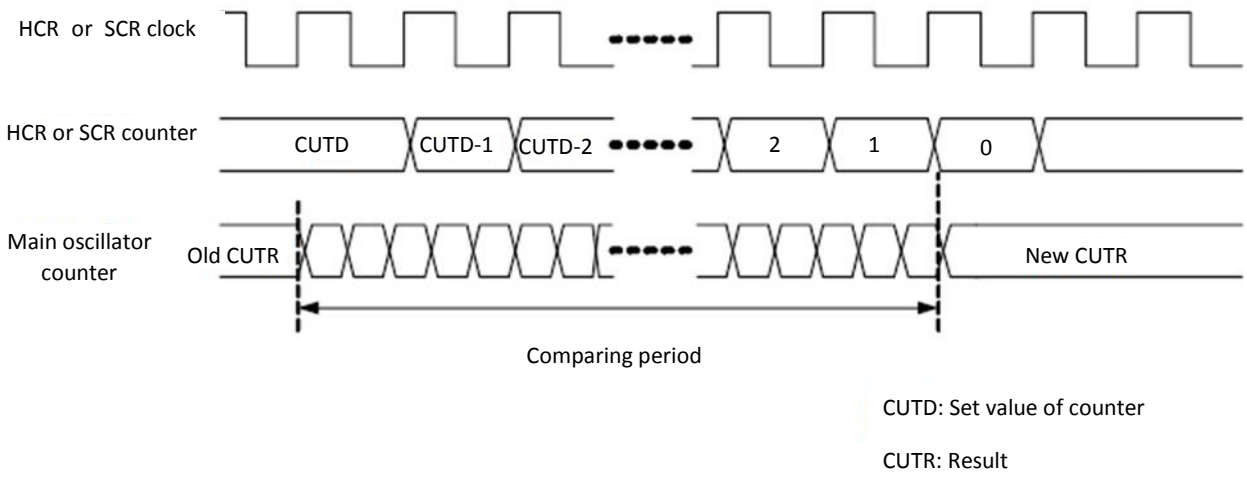
S6J3360/S6J3370/S6J3400/S6J3510 Series



2.2 Hardware Operation

The CR calibrator measures the count of the main clock when the main clock is being compared with the input clock. Figure 2 shows the timing diagram. HCR is the built-in high-speed CR oscillator, and SCR is the built-in slow-speed CR oscillator.

Figure 2. Timing Diagram of HCR or SCR Clock and Counters



CUTD: Set value of counter

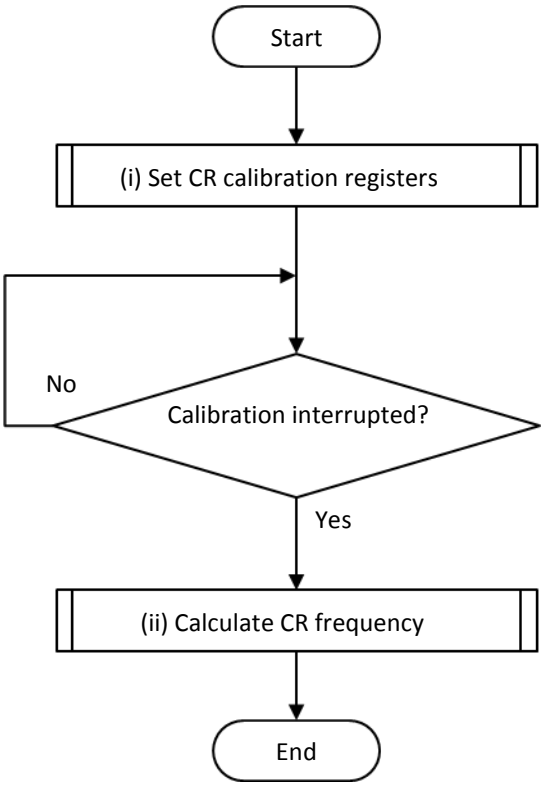
CUTR: Result

3 Built-In CR Clock Calibration

3.1 CR Clock Frequency Measurement and Calculation

Figure 3 shows the flowchart for the measurement and calculation of the HCR and SCR frequency. This procedure is used to calibrate the frequency of the HCR and SCR as described in the CR Clock Frequency Calibration section. The following sections provide details of each procedure in Figure 3.

Figure 3. Flow Chart of CR Clock Frequency Measurement and Calculation



(i) Set the CR calibration registers.

1. Set CR clock timer data register 1 (CU_CUTD1).
2. Set the interrupt enable bit (CU_CUCR1.INTEN) to '1'.
3. Set the correction start bit (CU_CUCR1.STRT) to '1'.

(ii) Calculate the CR frequency.

1. Read main oscillation timer data register 1 (CU_CUTR1).
2. Calculate the CR frequency using Equation 1.

Equation 1

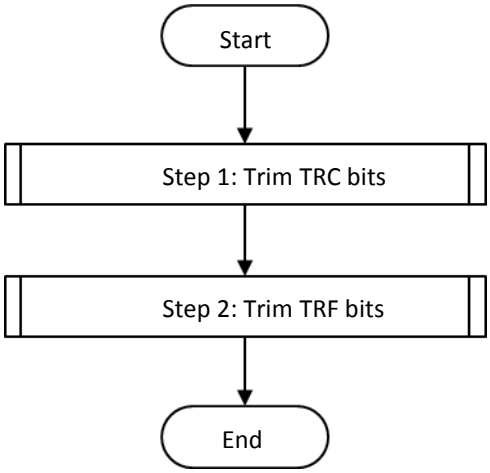
$$\text{CR frequency[MHz]} = \frac{\text{Main clock frequency[MHz]} \times \text{CU\_CUTD1}}{\text{CU\_CUTR1}}$$

3.2 CR Clock Frequency Calibration

Figure 4 shows the flowchart for the calibration of the HCR and SCR frequency. This procedure includes two steps:

1. Calibration with the TRC bits for low-resolution trimming
2. Calibration with the TRF bits for high-resolution trimming

Figure 4. Flow Chart of CR Clock Frequency Calibration



These steps are different for a product and a clock, as described in Table 2.

Table 2. CR oscillator type

Product Model and Clock	Step No.	Reference Section
S6J3110/S6J3120/S6J3200 HCR	Step 1 – TRC Bits Trimming	3.2.1
	Step 2 – TRF Bits Trimming	3.2.2
S6J3300/S6J3350/ S6J3360/S6J3370/S6J3400/S6J3510 HCR	Step 1 – TRC Bits Trimming	3.2.3
	Step 2 – TRF Bits Trimming	3.2.4
S6J3360/S6J3370/S6J3400/S6J3510 SCR	Step 1 – TRC Bits Trimming	3.2.5
	Step 2 – TRF Bits Trimming	3.2.6

3.2.1 S6J3110/S6J3120/S6J3200 for HCR: Step 1 – TRC Bits Trimming

Figure 5 shows the flowchart for the calibration with the TRC bits. The protection key should be set to lock release (SYSC0_PROTKEYR="0x5CACCE55") each time and just before the TRC and TRF bits are set. The following is steps of each procedure in Figure 5.

1. Set 5'b11111 in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
2. Set 5'b00000 in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_CA.
4. Set 5'b11111 in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
5. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_CB.
6. Calculate the TRC code using Equation 2, Equation 3, and Equation 4. The result is C_TRC.

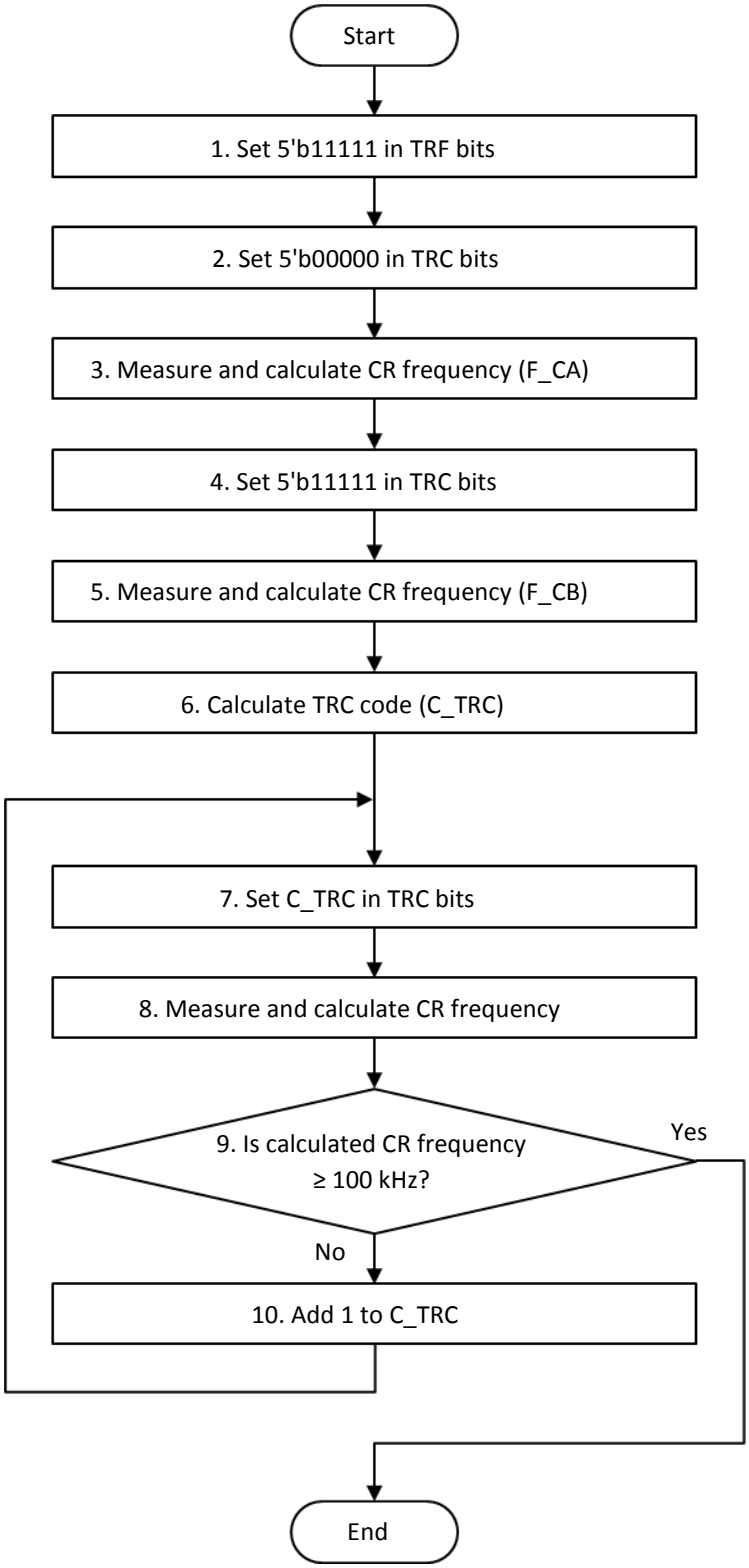
   Equation 2 $$T\_CA[us] = \frac{1}{F\_CA[MHz]}$$

   Equation 3 $$T\_CB[us] = \frac{1}{F\_CB[MHz]}$$

   Equation 4 $$C\_TRC = \frac{(T\_CA - 10) \times 31}{(T\_CA - T\_CB)}$$

   Note: The TRC code is rounded down to an integer.
7. Set C_TRC in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
8. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation).
9. If the calculated CR frequency is greater than or equal to 100 kHz, proceed to S6J3110/S6J3120/S6J3200 for HCR: Step 2 – TRF Bits Trimming. If the calculated CR frequency is less than 100 kHz, proceed to the next activity.
10. Add 1 to C_TRC and repeat from step 7.

Figure 5. Flow Chart of TRC Bits Trimming



3.2.2 S6J3110/S6J3120/S6J3200 for HCR: Step 2 – TRF Bits Trimming

Figure 6 shows the flowchart for the rough calibration with the TRF bits. The protection key should be set to lock release (SYSC0_PROTKEYR="0x5CACCE55") each time and just before the TRC and TRF bits are set. Steps of each procedure in Figure 6 are given below:

1. Set the code calculated in S6J3110/S6J3120/S6J3200 for HCR: Step 1 – TRC Bits Trimming (C_TRC) in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
2. Set 5'b00000 in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_FA.
4. Set 5'b11111 in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
5. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_FB.
6. Calculate the TRF code using Equation 5, $T\_FA[us] = \frac{1}{F\_FA[MHz]}$

   Equation 6, and Equation 7. The result is C_TRF_A.

   Equation 5 $$T\_FA[us] = \frac{1}{F\_FA[MHz]}$$

   Equation 6 $$T\_FB[us] = \frac{1}{F\_FB[MHz]}$$

   Equation 7 $$C\_TRF\_A = \frac{(T\_FA - 10) \times 31}{(T\_FA - T\_FB)}$$ **Note:** The TRF code is rounded down to an integer.
8. Set C_TRF_A in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
9. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_RA.
10. If the calculated CR frequency is greater than or equal to 100 kHz, proceed to (i) F_RA ≥ 100 kHz (see Figure 7). If the calculated CR frequency is less than 100 kHz, proceed to (ii) F_RA < 100 kHz (see Figure 7).

Figure 7 shows the flowchart for the fine calibration with the TRF bits. Steps of each procedure in (i) and (ii) are given below:

(i) F_RA ≥ 100 kHz

1. Subtract 1 from C_TRF_A. The result is C_TRF_B.
2. Set C_TRF_B in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_RB.
4. If | F_RA – 100 kHz | ≤ | F_RB – 100 kHz | is true, set C_TRF_A code and complete.
   If | F_RA – 100 kHz | > | F_RB – 100 kHz | is true, proceed to the next activity.
5. Replace C_TRF_A with C_TRF_B and F_RA with F_RB. Then repeat from activity 1.

(ii) F_RA < 100 kHz

1. Add 1 to C_TRF_A. The result is C_TRF_B.
2. Set C_TRF_B in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_RB.

4. If | 100 kHz – F_RA | ≤ | 100 kHz – F_RB | is true, set the calculated code and complete.
   If | 100 kHz – F_RA | > | 100 kHz – F_RB | is true, proceed to the next activity.
5. Replace C_TRF_A with C_TRF_B and F_RA with F_RB. Then repeat from activity 1.

Figure 6. Flow Chart of TRF Bits Trimming (Rough Calibration)

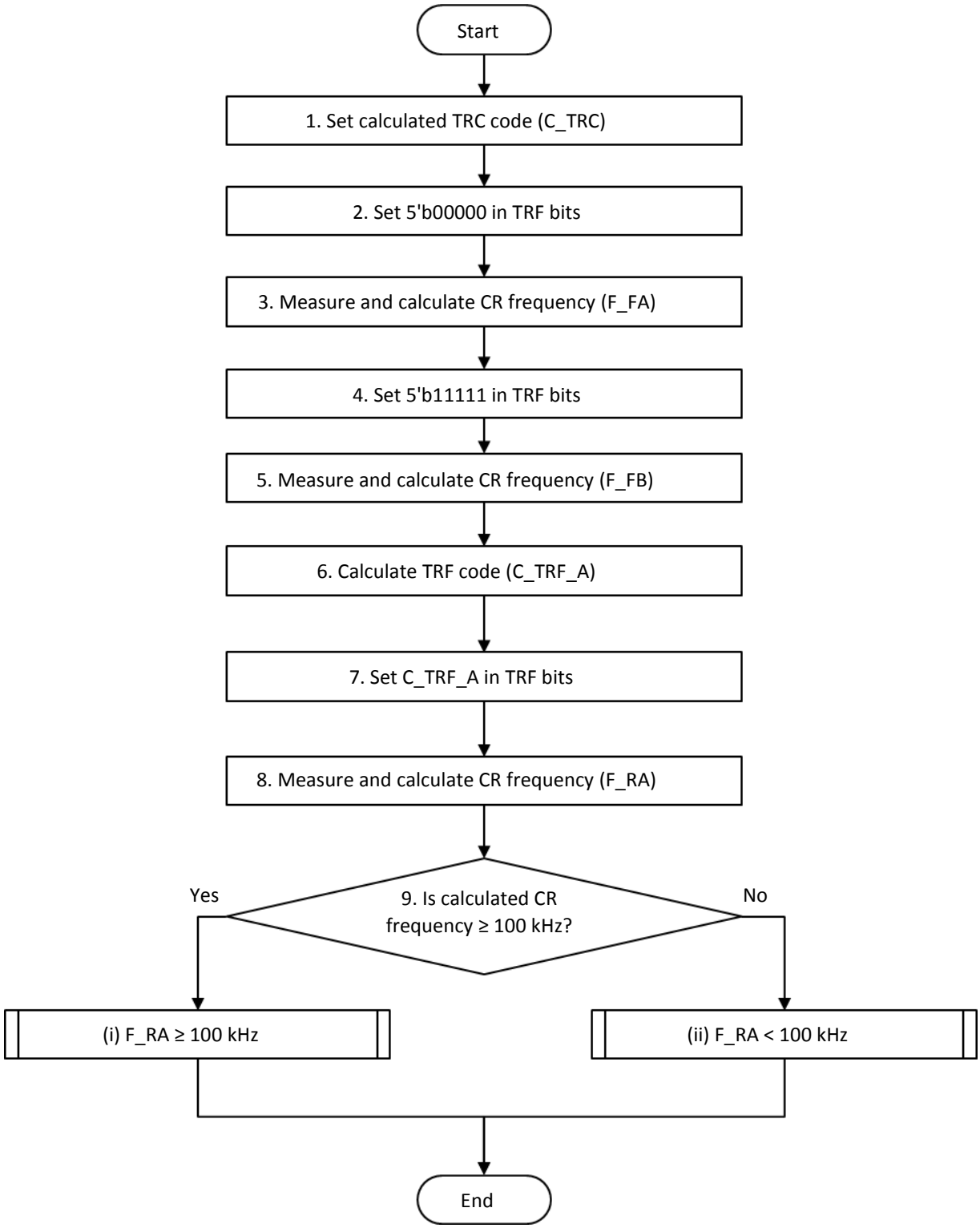
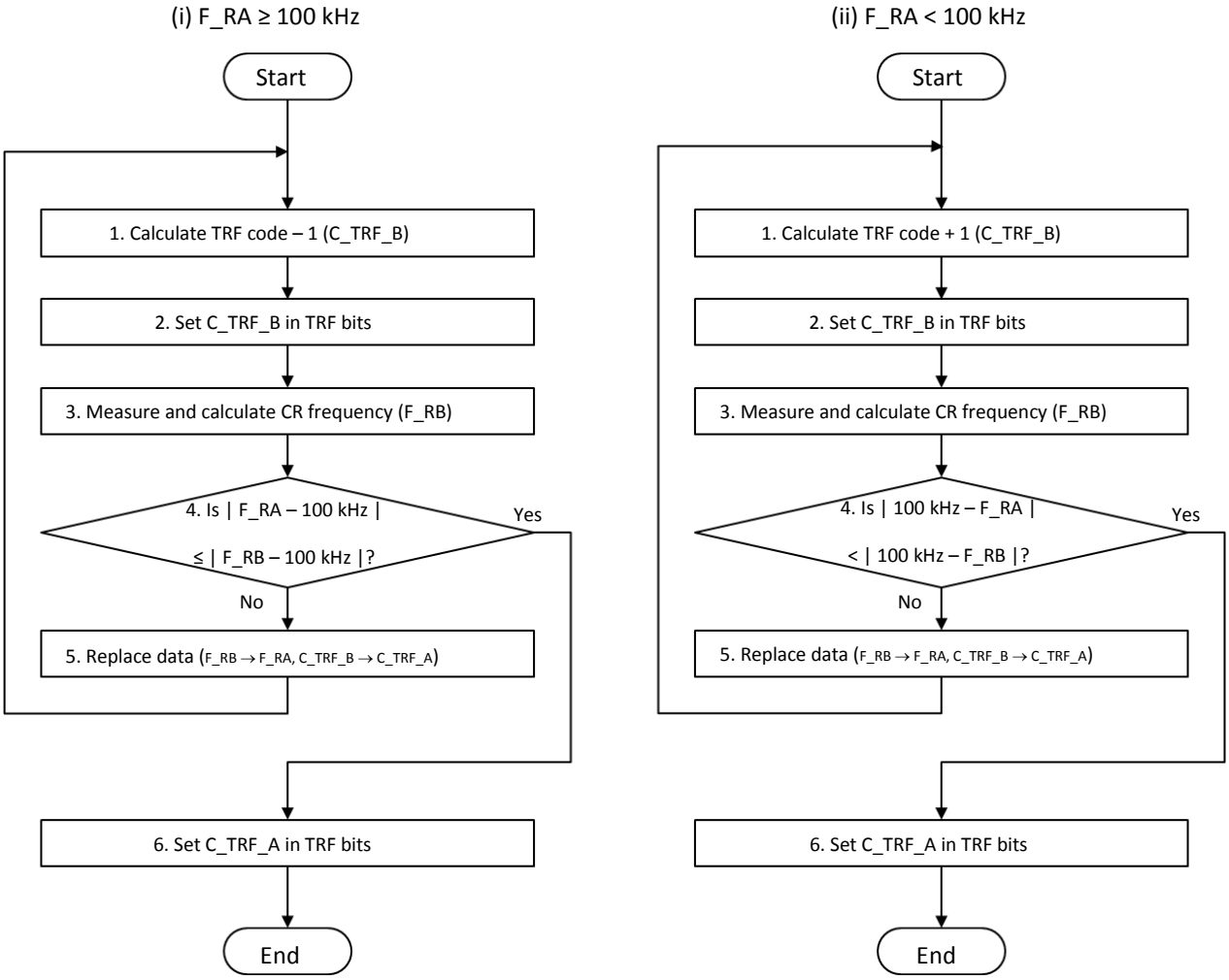


Figure 7. Flow Chart of TRF Bits Trimming (Fine Calibration)



3.2.3 S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 for HCR: Step 1 – TRC Bits Trimming

Figure 8 shows the flowchart for the calibration with the TRC bits. The protection key should be set to lock release (SYSC0_PROTKEYR="0x5CACCE55") each time and just before the TRC and TRF bits are set. Steps of each procedure in Figure 8 are given below:

1. Set 5'b00000 in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
2. Set 5'b10000 in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_CA.
4. Set 5'b10001 in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
5. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_CB.
6. Calculate the TRC code using Equation 8, Equation 9, and Equation 10. The result is C_TRC.

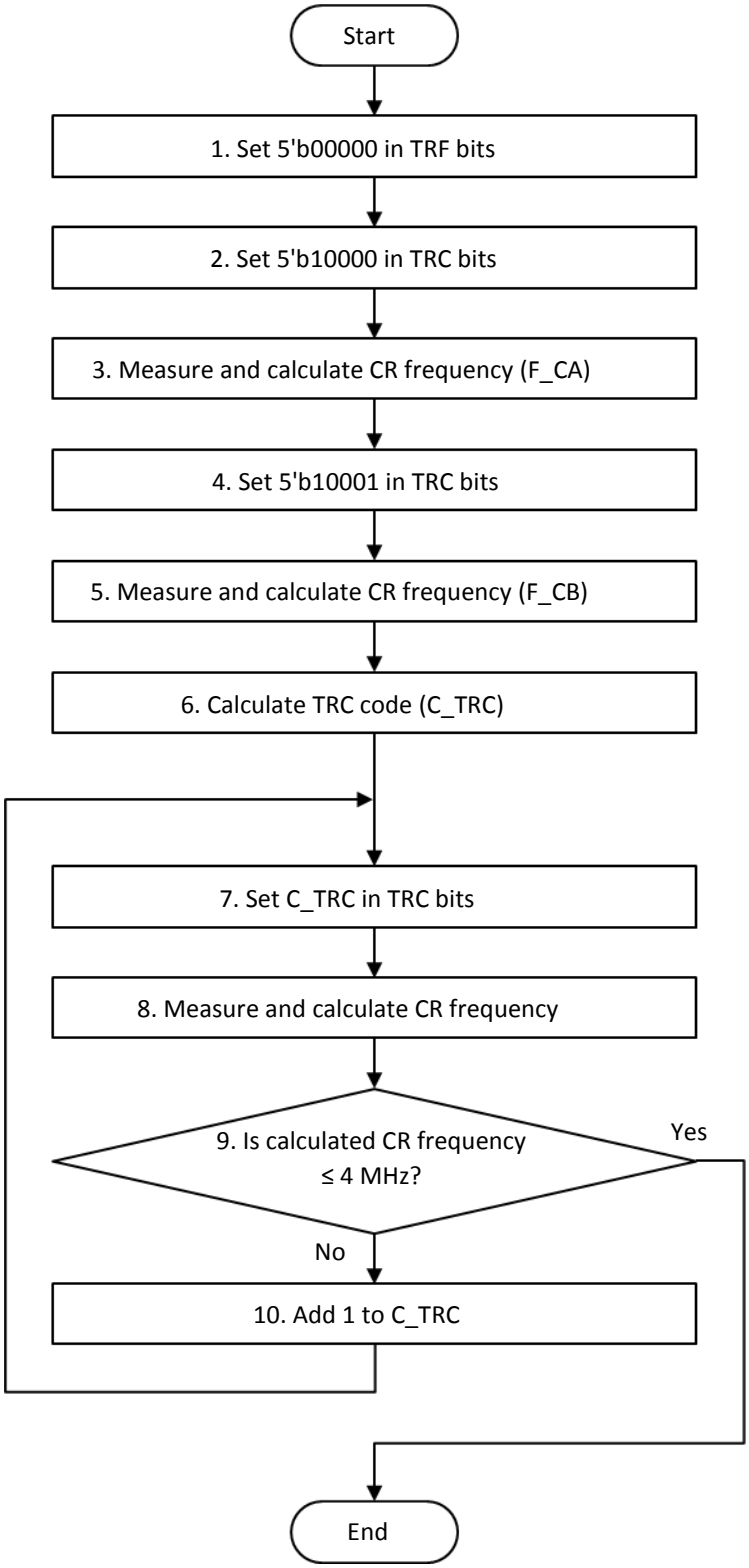
   Equation 8 $$T\_CA[us] = \frac{1}{F\_CA[MHz]}$$

   Equation 9 $$T\_CB[us] = \frac{1}{F\_CB[MHz]}$$

   Equation 10 $$C\_TRC = \left(16 + \frac{(0.25 - T\_CA)}{(T\_CB - T\_CA)}\right)$$

   Note: The TRC code is rounded down to an integer.
7. Set C_TRC in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
8. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation).
9. If the calculated CR frequency is less than or equal to 4 MHz, proceed to S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 for HCR: Step 2 – TRF Bits Trimming. If the calculated CR frequency is greater than 4 MHz, proceed to the next activity.
10. Add 1 to C_TRC and repeat from step 7.

Figure 8. Flow Chart of TRC Bits Trimming



3.2.4 S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 for HCR: Step 2 – TRF Bits Trimming

Figure 9 shows the flowchart for the rough calibration with the TRF bits. The protection key should be set to lock release (SYSC0_PROTKEYR="0x5CACCE55") each time and just before the TRC and TRF bits are set. Steps of each procedure in Figure 9 are given below:

1. Set the code calculated in S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 for HCR: Step 1 – TRC Bits Trimming (C_TRC) in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
2. Set 5'b00000 in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_FA.
4. Set 5'b11111 in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
5. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_FB.
6. Calculate the TRF code using Equation 11, Equation 12, and Equation 13. The result is C_TRF.

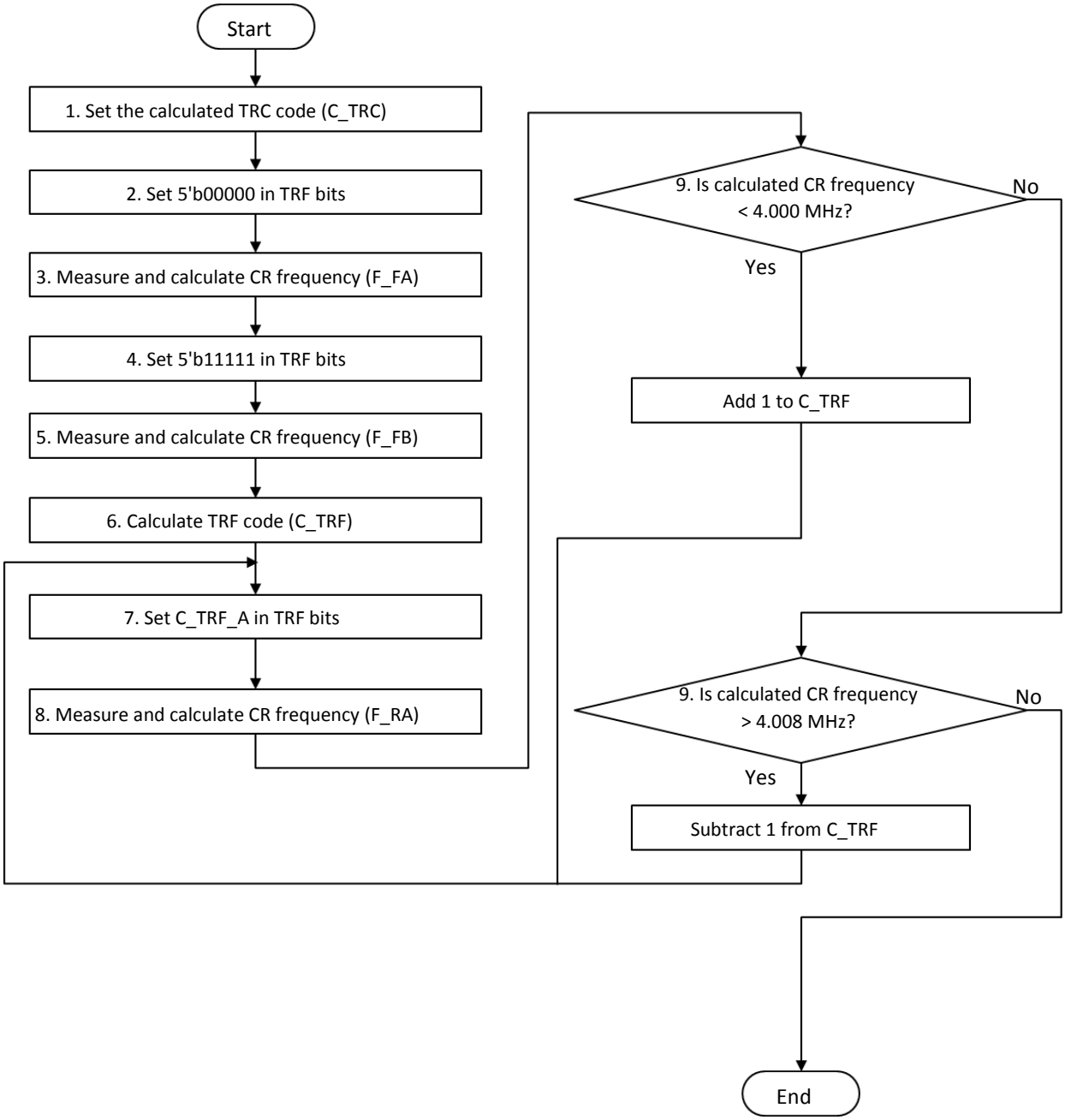
   Equation 11 $$T\_FA[us] = \frac{1}{F\_FA[MHz]}$$

   Equation 12 $$T\_FB[us] = \frac{1}{F\_FB[MHz]}$$

   Equation 13 $$C\_TRF = \left(\frac{(0.25 - T\_FA)}{((T\_FB - T\_FA)/31)}\right)$$ **Note:** The TRF code is rounded up to an integer.

7. Set C_TRF_A in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
8. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_RA.
9. If the calculated CR frequency is less than 4.000 MHz, add 1 to C_TRC and repeat from step 7.
   If the calculated CR frequency is greater than 4.008 MHz, subtract 1 from C_TRC and repeat from step 7.

Figure 9. Flow Chart of TRF Bits Trimming (Rough Calibration)



3.2.5 S6J3360/S6J3370/S6J3400/S6J3510 for SCR: Step 1 – TRC Bits Trimming

Figure 10 shows the flowchart for the calibration with the TRC bits. The protection key should be set to lock release (SYSC0_PROTKEYR="0x5CACCE55") each time and just before the TRC and TRF bits are set. Steps of each procedure in Figure 10 are given below:

1. Set 4'b1111 in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
2. Set 3'b000 in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_CA.
4. Set 3'b111 in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
5. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_CB.
6. Calculate the TRC code using Equation 14, Equation 15, and Equation 16. The result is C_TRC.

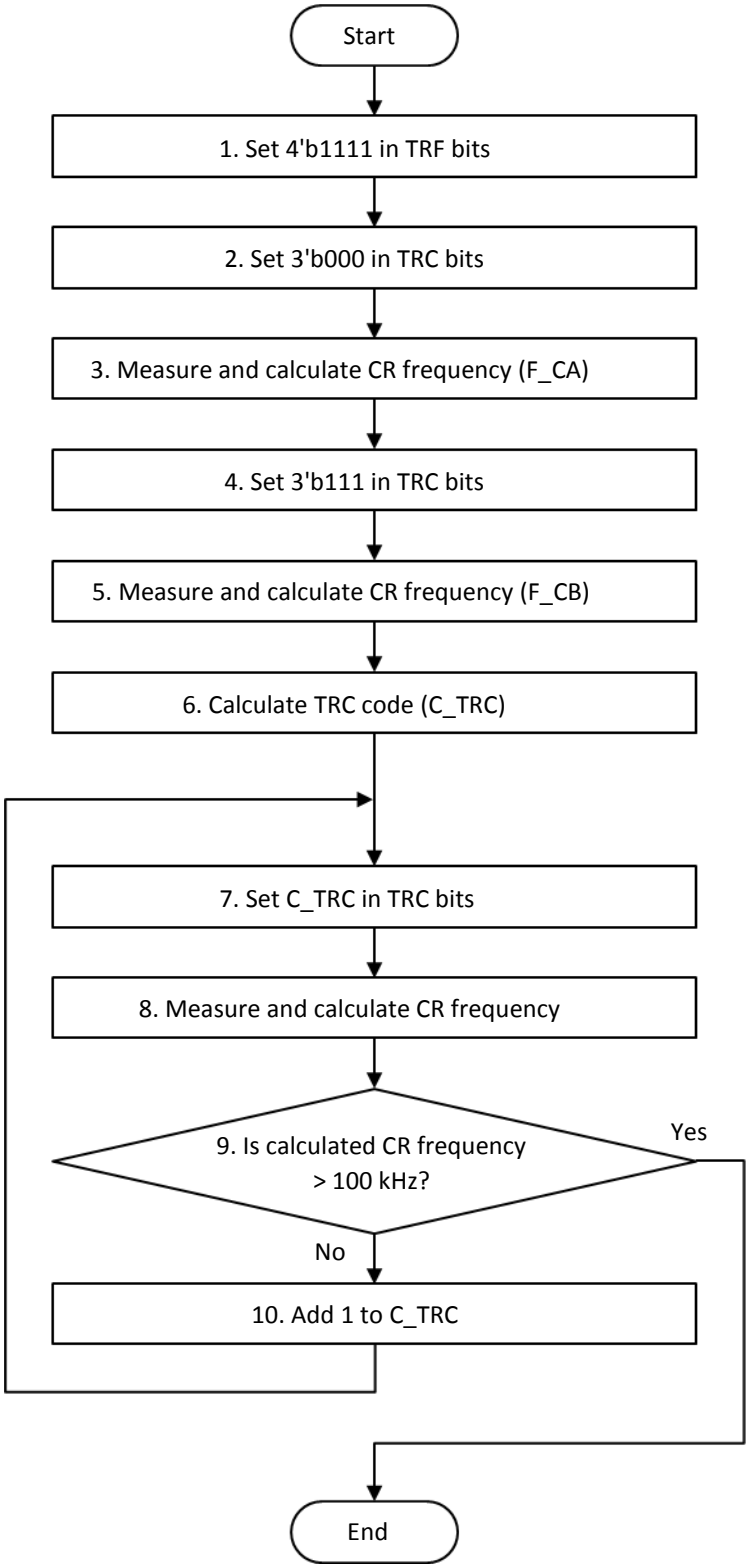
   Equation 14 $$T\_CA[us] = \frac{1}{F\_CA[MHz]}$$

   Equation 15 $$T\_CB[us] = \frac{1}{F\_CB[MHz]}$$

   Equation 16 $$C\_TRC = \left(\frac{(T\_CA - 10)}{((T\_CA - T\_CB)/7)}\right)$$

   Note: The TRC code is rounded down to an integer.
7. Set C_TRC in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
8. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation).
9. If the calculated CR frequency is greater than 100 kHz, proceed to S6J3360/S6J3370/S6J3400/S6J3510 for SCR: Step 2 – TRF Bits Trimming. If the calculated CR frequency is less than 100 kHz, proceed to the next activity.
10. Add 1 to C_TRC and repeat from step 7.

Figure 10. Flow Chart of TRC Bits Trimming



3.2.6 S6J3360/S6J3370/S6J3400/S6J3510 for SCR: Step 2 – TRF Bits Trimming

Figure 11 shows the flowchart for the rough calibration with the TRF bits. The protection key should be set to lock release (SYSC0_PROTKEYR="0x5CACCE55") each time and just before the TRC and TRF bits are set. Steps of each procedure in Figure 11 are given below:

1. Set the code calculated in S6J3360/S6J3370/S6J3400/S6J3510 for SCR: Step 1 – TRC Bits Trimming (C_TRC) in the TRC bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
2. Set 4'b0000 in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_FA.
4. Set 4'b1111 in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
5. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_FB.
6. Calculate the TRF code using Equation 17, Equation 18, and Equation 19. The result is C_TRF_A.

   Equation 17 $$\mathrm{T\_FA[us]} = \frac{1}{\mathrm{F\_FA[MHz]}}$$

   Equation 18 $$\mathrm{T\_FB[us]} = \frac{1}{\mathrm{F\_FB[MHz]}}$$

   Equation 19 $$\mathrm{C\_TRF\_A} = \frac{(\mathrm{T\_FA} - 10)}{((\mathrm{T\_FA} - \mathrm{T\_FB})/15)}$$

   Note: The TRF code is rounded down to an integer.
7. Set C_TRF_A in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
8. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_RA.
9. If the calculated CR frequency is greater than or equal to 100 kHz, proceed to (i) F_RA ≥ 100 kHz (see Figure 12). If the calculated CR frequency is less than 100 kHz, proceed to (ii) F_RA < 100 kHz (see Figure 12).

Figure 12 shows the flowchart for the fine calibration with the TRF bits. The following are details of each procedure in (i) and (ii) .

(i) F_RA ≥ 100 kHz

1. Subtract 1 from C_TRF_A. The result is C_TRF_B.
2. Set C_TRF_B in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_RB.
4. If | F_RA – 100 kHz | ≤ | F_RB – 100 kHz | is true, set C_TRF_A code and complete.
   If | F_RA – 100 kHz | > | F_RB – 100 kHz | is true, proceed to the next activity.
5. Replace C_TRF_A with C_TRF_B and F_RA with F_RB. Then repeat from activity 1.

(ii) F_RA < 100 kHz

1. Add 1 to C_TRF_A. The result is C_TRF_B.
2. Set C_TRF_B in the TRF bits after setting the protection key to lock release (SYSC0_PROTKEYR="0x5CACCE55").
3. Measure and calculate the CR frequency (refer to CR Clock Frequency Measurement and Calculation). The result is F_RB.
4. If | 100 kHz – F_RA | ≤ | 100 kHz – F_RB | is true, set the calculated code and complete.
   If | 100 kHz – F_RA | > | 100 kHz – F_RB | is true, proceed to the next activity.
5. Replace C_TRF_A with C_TRF_B and F_RA with F_RB. Then repeat from activity 1.

Figure 11. Flow Chart of TRF Bits Trimming (Rough Calibration)

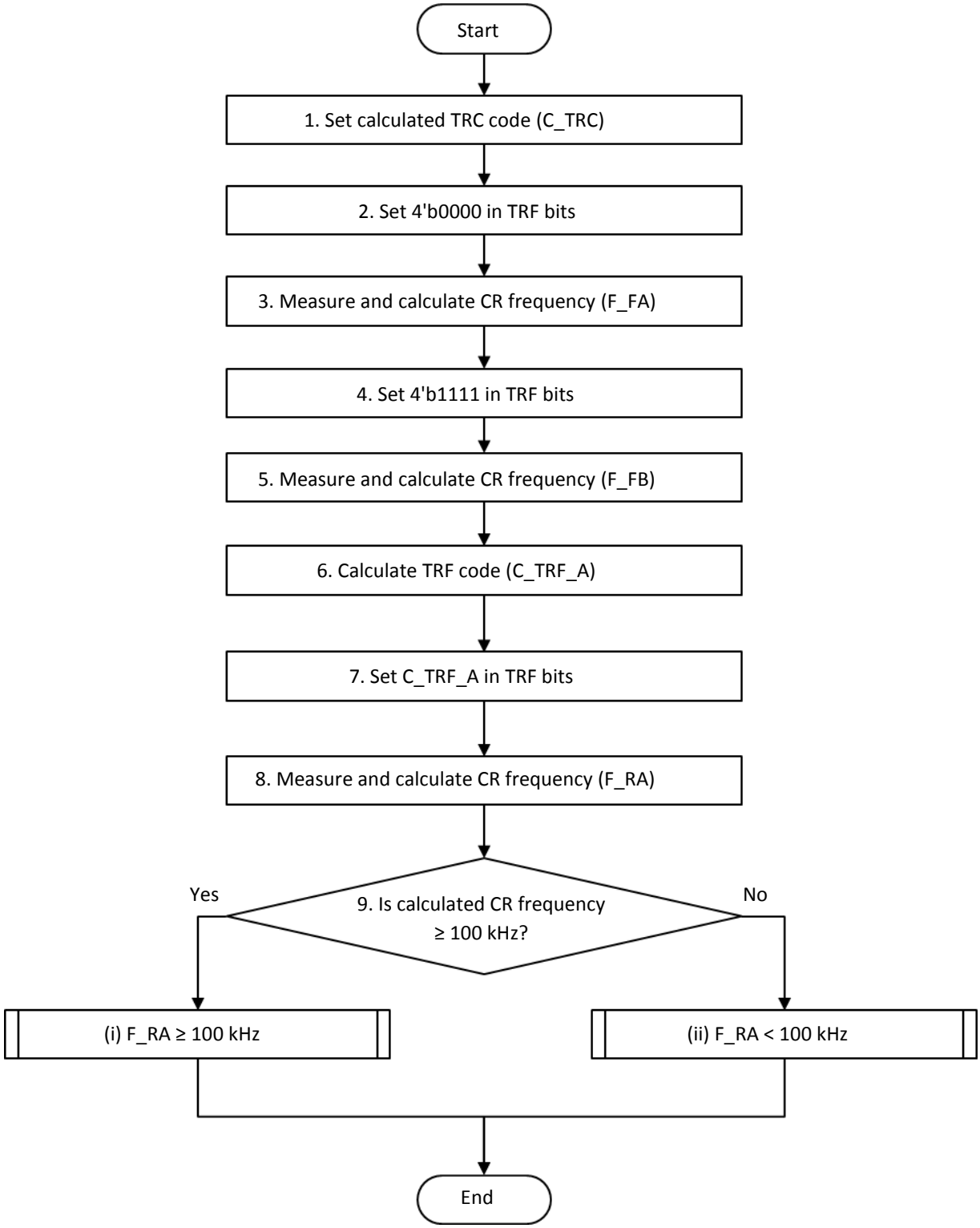
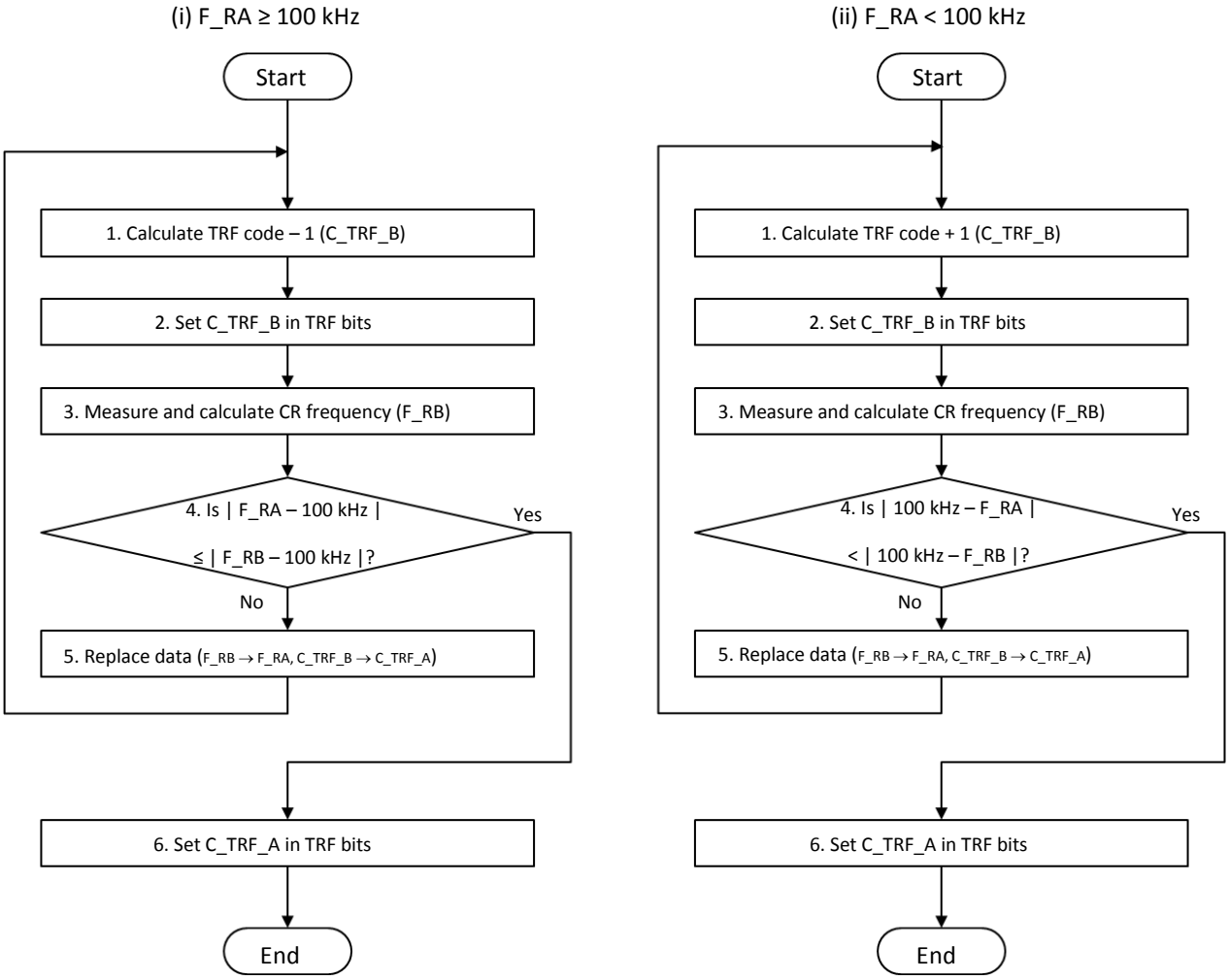


Figure 12. Flow Chart of TRF Bits Trimming (Fine Calibration)



4 Related Documents

- S6J311E/D/C/B Series Datasheet (Doc. No.002-05681)
- S6J311A/9/8 Series Datasheet (Doc. No.002-04632)
- S6J3110 Series Hardware Manual (Doc.No.002-10667)
- S6J3120 Series Datasheet (Doc.No.002-04863)
- S6J3120 Series Hardware Manual (Doc.No.002-04855)
- S6J3200 Series Datasheet (Doc.No.002-05682)
- S6J3200 Series Hardware Manual (Doc.No.002-04852)
- S6J32E/F/G Series Datasheet (Doc.No.002-10689)
- S6J32E/F/G Series Hardware Manual (Doc.No.002-12500)
- Traveo Family Hardware Manual Platform Part for S6J3200 Series (Doc.No.002-04854)
- S6J3310/20/30/40 Series Datasheet (Doc.No.002-10635)
- S6J3350 Series Datasheet (Doc.No.002-10634)
- S6J3310/20/30/40/50 Series Hardware Manual (Doc.No.002-10185)
- Traveo Family Hardware Manual Platform Part for S6J3310/3320/3330/3340/3350 Series (Doc.No.002-07884)
- S6J3360/70 Series Datasheet (Doc.No.002-03359)
- S6J3360/70 Series Hardware Manual (Doc.No.002-18302)
- Traveo Family Hardware Manual Platform Part for S6J3360/3370 Series (Doc.No.002-07884)
- S6J3400 Series Datasheet (Doc.No.001-97829)
- S6J3400 Series Hardware Manual (Doc.No.002-09919)
- Traveo Family Hardware Manual Platform Part for S6J3400 Series (Doc.No.002-07884)
- S6J3510 Series Datasheet (Doc.No.002-18647)
- S6J3510 Series Hardware Manual (Doc.No.002-18642)
- Traveo Family Hardware Manual Platform Part for S6J3510 Series (Doc.No.002-07884)

5 Summary

This application note described how the CR calibration function enables you to calibrate the built-in CR clock.

Appendix A. S6J3110/S6J3120/S6J3200 Series Example Program for HCR

This appendix shows an example program of the S6J3110/S6J3120/S6J3200 series for HCR. You can modify this code for other products and clocks.

A.1 Example Program of Constants and Parameters

Code 1 describes the example program of constants and parameters.

Code 1. Constants and Parameters

```
#define TR_MAX       0x1F
#define TR_MIN       0x00

#define UNLOCK_KEY   0x5CACCE55

#define ABS(a, b) ((a) >= (b) ? (a-b) : (b-a)) // calculation of absolute value

#define CCM   1000000       // Calculation Coefficient 10^6
#define CCK   1000          // Calculation Coefficient 10^3

unsigned char calibration_state;  // calibration procedure number

unsigned char fb_calc_mode;       // selected calculation mode

unsigned long freq_calc_result;
unsigned long f_ca;
unsigned long f_cb;
unsigned long t_ca;
unsigned long t_cb;
unsigned long c_trc;
unsigned long f_tmp;
unsigned long t_tmp;
unsigned long f_fa;
unsigned long f_fb;
unsigned long t_fa;
unsigned long t_fb;
unsigned long c_trf_a;
unsigned long c_trf_b;
unsigned long f_ra;
unsigned long f_rb;
unsigned long t_ra;
unsigned long t_rb;

unsigned char flag_measure_complete;    // software flag for CR measurement
```

Constants for measurement of TRC code and TRF code

Unlock key for the register SYSC_CRCNTR

Parameters for calculation of frequency, TRC code, and TRF code

A.2 Example Program of CR Clock Frequency Measurement and Calculation

Code 2 describes the example program of CR clock frequency measurement and calculation. The details are given in CR Clock Frequency Measurement and Calculation.

Code 2. CR Clock Frequency Measurement and Calculation

(i) of Figure 3

```
static void StartMeasureCR(void)
{
    flag_measure_complete = 0;    // reset measure complete flag (software flag)

    CU_CUTD1 = 200;               // set CR measurement duration
    CU_CUCR1 = 0x0011;            // start CR measure
}
```

Arbitrary value

(ii) of Figure 3

Interrupt function

```
FN_IRQ_DEFINE_BEGIN(cu_interrupt, INTERRUPTS_IRQ_NUMBER_117)
{
    unsigned long    comp_count;
    unsigned long    cr_count;

    CU_CUCRC1 = 0x0002;           // clear interrupt flag

    comp_count = CU_CUTR1;        // read main oscillation timer count
    cr_count   = CU_CUTD1;        // read CR oscillation timer count

    /* calculate frequency */
    freq_calc_result = ((4*cr_count)*CCM) / comp_count;

    flag_measure_complete = 1;    // set measure complete flag (software flag)
}
FN_IRQ_DEFINE_END()
```

Multiply by the coefficient so that a result won't be less than 0.

A.3 Example Program of CR Clock Frequency Calibration

Code 3 describes the example program of CR clock frequency calibration. The details are given in CR Clock Frequency Calibration.

Code 3. CR Clock Frequency Calibration

```
int main(void)
{
    // Finalize initialization to default settings.
    // (this will do IRQ and NMI initialization and global IRQ/NMI enable)
    Start_Init();

    calibration_state = 0;

    // Endless loop
    for(;;)
    {
        ClearWatchdog();

        switch(calibration_state)
        {
        case 0:
            /**************/
            /*** step 1 ***/
            /**************/

            /* set TRF=5'b11111 */
            SYSC0_PROTKEYR  = UNLOCK_KEY; // unlock register protect
            SYSC_CRCNTR_TRF = TR_MAX;

            /* set TRC=5'b00000 */
            SYSC0_PROTKEYR  = UNLOCK_KEY; // unlock register protect
            SYSC_CRCNTR_TRC = TR_MIN;

            calibration_state++;          // forward state

            /* measure CR */
            StartMeasureCR();
            break;
        case 1:
            /* check measure complete? */
            if(flag_measure_complete==1)
            {
                f_ca = freq_calc_result;  // set calculation result
                calibration_state++;      // forward state
            }
            break;
        case 2:
            /* set TRC=5'b11111 */
            SYSC0_PROTKEYR = UNLOCK_KEY;  // unlock register protect
            SYSC_CRCNTR_TRC = TR_MAX;

            calibration_state++;          // forward state

            /* measure CR */
            StartMeasureCR();
            break;
```

1 of step 1

2 of step 1

3 of step 1

4 of step 1

5 of step 1

5 of step 1

```
case 3:
    /* check measure complete? */
    if(flag_measure_complete==1)
    {
        f_cb = freq_calc_result;    // set calculation result
        calibration_state++;        // forward state
    }
    break;
```

6 of step 1

```
case 4:
    /* calculate TRC code */
    t_ca  = (1*CCM*CCK) / f_ca;
    t_cb  = (1*CCM*CCK) / f_cb;
    c_trc = (t_ca-(10*CCK))/((t_ca-t_cb)/31);
```

Multiply by the coefficient so that a result won't be less than 0.

7 of step 1 and 1 of step 2

```
    /* set TRC=c_trc */
    SYSC0_PROTKEYR = UNLOCK_KEY;   // unlock register protect
    SYSC_CRCNTR_TRC = c_trc;

    calibration_state++;           // forward state
```

8 of step 1

```
    /* measure CR */
    StartMeasureCR();
    break;
case 5:
    /* check measure complete? */
    if(flag_measure_complete==1)
    {
        f_tmp = freq_calc_result;         // set calculation result
        t_tmp = (1*CCM*CCK) / f_tmp;      // calculate frequency time

        calibration_state++;              // forward state
    }
    break;
```

Multiply by the coefficient so that a result won't be less than 0.

9 and 10 of step 1

```
case 6:
    if(t_tmp>=10*CCK)
    {
        calibration_state++;        // forward state (go to step 2)
    }
    else
    {
        c_trc = c_trc + 1;
        /* calculate TRC code */
        SYSC0_PROTKEYR = UNLOCK_KEY;      // unlock register protect
        SYSC_CRCNTR_TRC = c_trc;

        calibration_state = 5;      // go back state

        /* measure CR */
        StartMeasureCR();
    }
    break;
```

Multiply by the coefficient so that a result won't be less than 0.

```
                case 7:
                    /**************/
                    /*** step 2 ***/
                    /**************/

2 of step 2         /* set TRF=5'b00000 */
                    SYSC0_PROTKEYR = UNLOCK_KEY;   // unlock register protect
                    SYSC_CRCNTR_TRF = TR_MIN;

                    calibration_state++;           // forward state

3 of step 2         /* measure CR */
                    StartMeasureCR();
                    break;
                case 8:
3 of step 2         /* check measure complete? */
                    if(flag_measure_complete==1)
                    {
                        f_fa = freq_calc_result;   // set calculation result
                        calibration_state++;       // forward state
                    }
                    break;
                case 9:
4 of step 2         /* set TRF=5'b11111 */
                    SYSC0_PROTKEYR = UNLOCK_KEY;   // unlock register protect
                    SYSC_CRCNTR_TRF = TR_MAX;

                    calibration_state++;           // forward state

                    /* measure CR */
                    StartMeasureCR();
                    break;
                case 10:
5 of step 2         /* check measure complete? */
                    if(flag_measure_complete==1)
                    {
                        f_fb = freq_calc_result;   // set calculation result
                        calibration_state++;       // forward state
                    }
                    break;
                case 11:
                    /* calculate TRF code */
6 of step 2         t_fa  = (1*CCM*CCK) / f_fa;
                    t_fb  = (1*CCM*CCK) / f_fb;
                    c_trf_a = (t_fa-(10*CCK))/((t_fa-t_fb)/31);

                    /* set TRF=c_trf_a */
7 of step 2         SYSC0_PROTKEYR = UNLOCK_KEY;   // unlock register protect
                    SYSC_CRCNTR_TRF = c_trf_a;

                    calibration_state++;           // forward state

                    /* measure CR */
8 of step 2         StartMeasureCR();
                    break;
```

Multiply by the coefficient so that a result won't be less than 0.

8 of step 2

```
case 12:
    /* check measure complete? */
    if(flag_measure_complete==1)
    {
        f_ra = freq_calc_result;          // set calculation result
        t_ra = (1*CCM*CCK) / f_ra;        // calculate frequency time

        calibration_state++;              // forward state
    }
    break;
```

Multiply by the coefficient so that a result won't be less than 0.

9 of step 2,

(i) of step 2,

and

(ii) of step 2

```
case 13:
    if(t_ra>=10*CCK)
    {
        c_trf_b = c_trf_a - 1;            // subtract 1 from c_trf_a

        /* set TRF=c_trf_b */
        SYSC0_PROTKEYR = UNLOCK_KEY;      // unlock register protect
        SYSC_CRCNTR_TRF = c_trf_b;
```

Multiply by the coefficient so that a result won't be less than 0.

9 of step 2,

(i) of step 2,

and

(ii) of step 2

```
        calibration_state++;       // forward state
        fb_calc_mode = 1;          // set calculation mode

        /* measure CR */
        StartMeasureCR();
    }
    else
    {
        c_trf_b = c_trf_a + 1;     // add 1 to c_trf_a

        /* set TRF=c_trf_b */
        SYSC0_PROTKEYR = UNLOCK_KEY;      // unlock register protect
        SYSC_CRCNTR_TRF = c_trf_b;

        calibration_state++;       // forward state
        fb_calc_mode = 2;          // set calculation mode

        /* measure CR */
        StartMeasureCR();
    }
    break;
```

(i) of step 2

and

(ii) of step 2

```
case 14:
    /* check measure complete? */
    if(flag_measure_complete==1)
    {
        f_rb = freq_calc_result;          // set calculation result
        t_rb = (1*CCM*CCK) / f_rb;        // calculate frequency time

        calibration_state++;              // forward state
    }
    break;
```

Multiply by the coefficient so that a result won't be less than 0.

```
case 15:
    if(fb_calc_mode==1)
    {
        if(ABS(t_ra, 10*CCK) <= ABS(t_rb, 10*CCK))
        {
            /* set TRF code (fixed) */
            SYSC0_PROTKEYR = UNLOCK_KEY;   // unlock register protect
            SYSC_CRCNTR_TRF = c_trf_a;
            calibration_state++;           // forward state (end)
        }
        else
        {
            /* copy data */
            t_ra = t_rb;
            c_trf_a = c_trf_b;

            c_trf_b = c_trf_a - 1;         // subtract 1 from c_trf_a

            /* set TRF=c_trf_b */
            SYSC0_PROTKEYR = UNLOCK_KEY;   // unlock register protect
            SYSC_CRCNTR_TRF = c_trf_b;

            calibration_state = 14;        // go back state

            /* measure CR */
            StartMeasureCR();
        }
    }
    else if(fb_calc_mode==2)
    {
        if(ABS(10*CCK, t_ra) <= ABS(10*CCK, t_rb))
        {
            /* set TRF code (fixed) */
            SYSC0_PROTKEYR = UNLOCK_KEY;   // unlock register protect
            SYSC_CRCNTR_TRF = c_trf_a;

            calibration_state++;           // forward state (end)
        }
        else
        {
            /* copy data */
            t_ra = t_rb;
            c_trf_a = c_trf_b;
            c_trf_b = c_trf_a + 1;

            /* set TRF=c_trf_b */
            SYSC0_PROTKEYR = UNLOCK_KEY;   // unlock register protect
            SYSC_CRCNTR_TRF = c_trf_b;

            calibration_state = 14;        // go back state

            /* measure CR */
            StartMeasureCR();
        }
    }
    break;
```

Multiply by the coefficient so that a result won't be less than 0.

Multiply by the coefficient so that a result won't be less than 0.

(i) of step 2

(ii) of step 2

(ii) of step 2

```
            case 16:
                calibration_state = 99;        // end state (calibration complete)
                break;
            default:
                break;
            }
        }
    }
```

Document History

Document Title: AN204096 - Built-In CR Clock Calibration for the Traveo™ Family

Document Number: 002-04096

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5016150	JUMA	11/27/2015	New application note
*A	5146949	JUMA	03/16/2016	Added target products S6J3200/S6J3300/S6J3350 Series
*B	5309163	JUMA	06/22/2016	Added target product S6J3400 Series Updated template
*C	5670746	HMIZ	05/26/2017	Added target product S6J3360/S6J3370/S6J3510 Series Updated template
*D	6332126	YOST	10/04/2018	No change. Sunset reviewed.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709