AN204455

How to Set Up the Clock System for the Traveo™ Family

Author: Kotaro Hachisu

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

Related Documents: For a complete list, see Related Documents.

This application note describes the clock setting procedure for the Traveo™ family S6J3110/3120/3200/3310/3320/3330/3340/3350/3360/3370/3400/3510 series 32-bit microcontrollers.

Contents

1 Introduction

This application note provides details on the clock settings for the Traveo family S6J3110/3120/3200/3310/3320/3330/3340/3350/3360/3370/3400/3510 series microcontrollers.

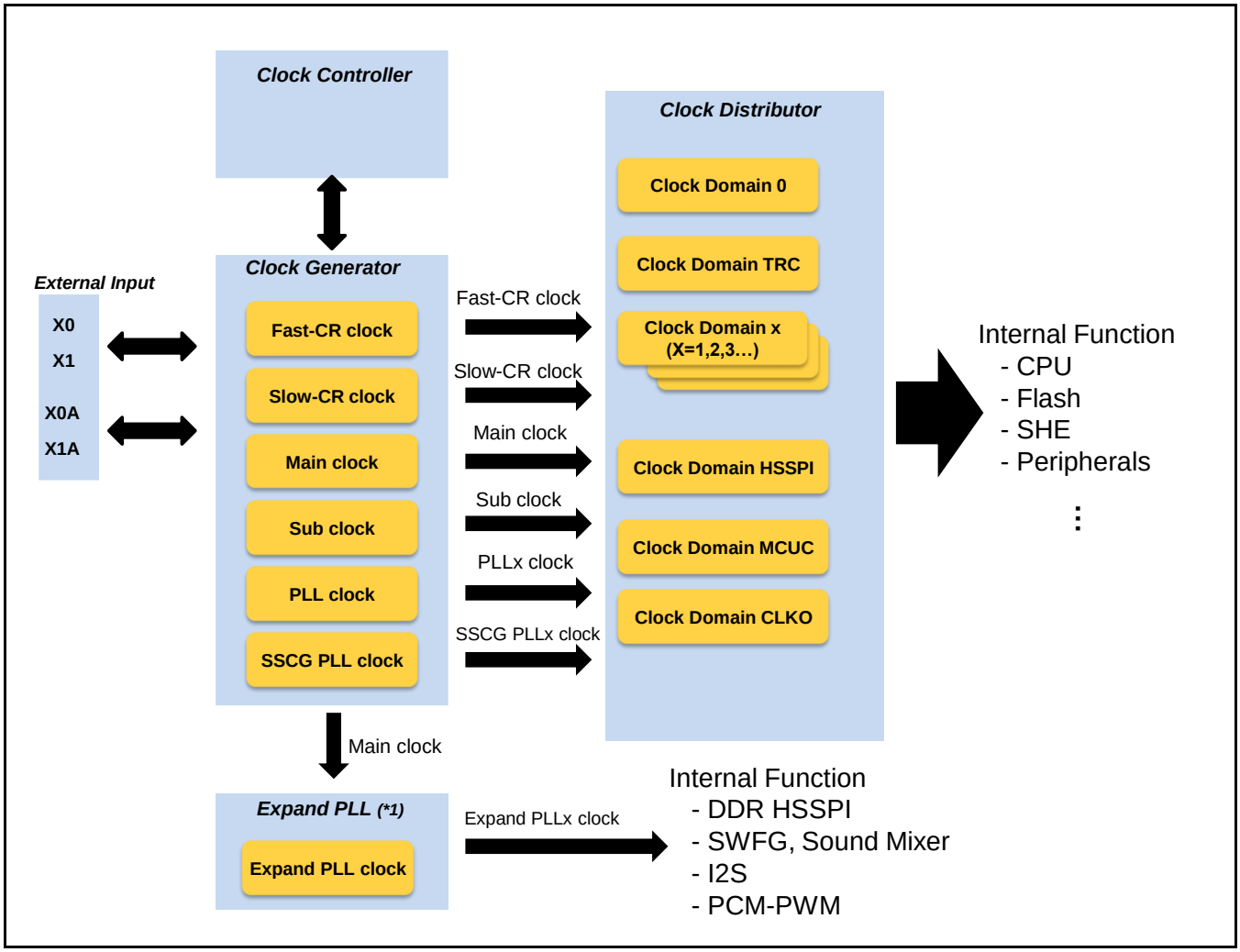
2 Overview

This section gives an overview of the clock system shown in Figure 1. The clock system is composed of four modules:

- Clock controller
- Clock generator
- Clock distributor
- Expand PLL [1]

[1] Expand PLL is only supported in S6J3360/ S6J3370/ S6J3510 series microcontrollers.

Figure 1. Clock System Block Diagram



2.1 Clock Controller

The clock controller generates a control signal from the configuration register value for the clock system.

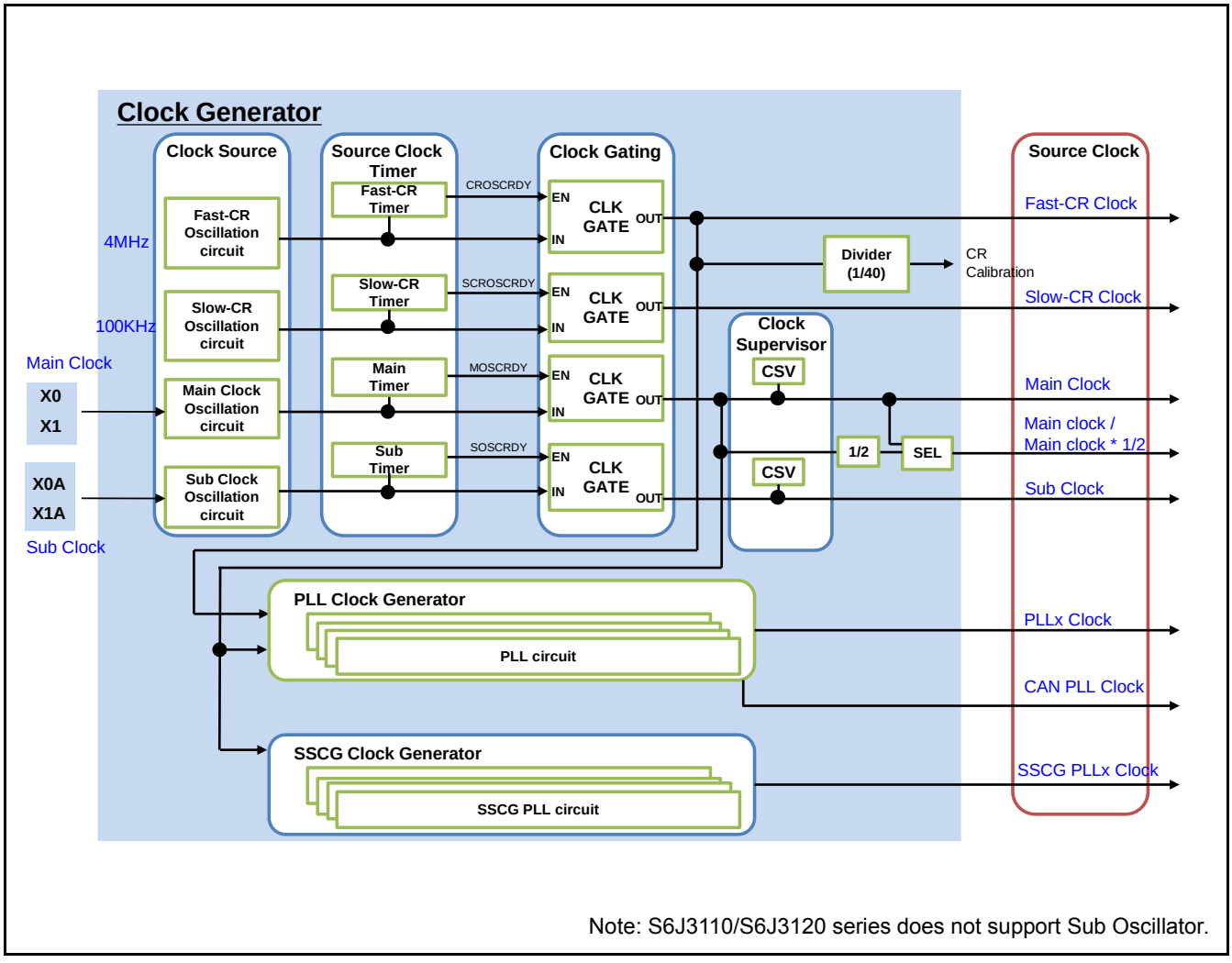
2.2 Clock Generator

The clock generator shown in Figure 2 generates the source clock from an oscillation circuit. The Traveo family microcontrollers support several types of clock sources:

- Main clock
- Fast-CR clock
- Slow-CR clock
- Sub clock

For the frequency of each clock, see the datasheet of the target products. All clock sources have a source clock timer for masking the clock output until the end of the clock oscillation stabilization wait time. For low-power operation, you can set the clock gating to allow a clock shutoff for the internal circuit.

Figure 2. Clock Generator Block Diagram



Note: S6J3110/S6J3120 series does not support Sub Oscillator.

The clock generator block has several PLL multiplier circuits. The main clock or fast-CR clock is the source of the PLL clock generator shown in Figure 3. When using the PLL clock, you need to select the PLL clock divider and multiplication by setting the appropriate bits, such as the PLLDIVL, PLLDIVN, and PLLDIVM bits, in the RUN PLLx control register. Table 1 gives an example of PLL clock configuration.

Figure 3. PLL Clock Generator

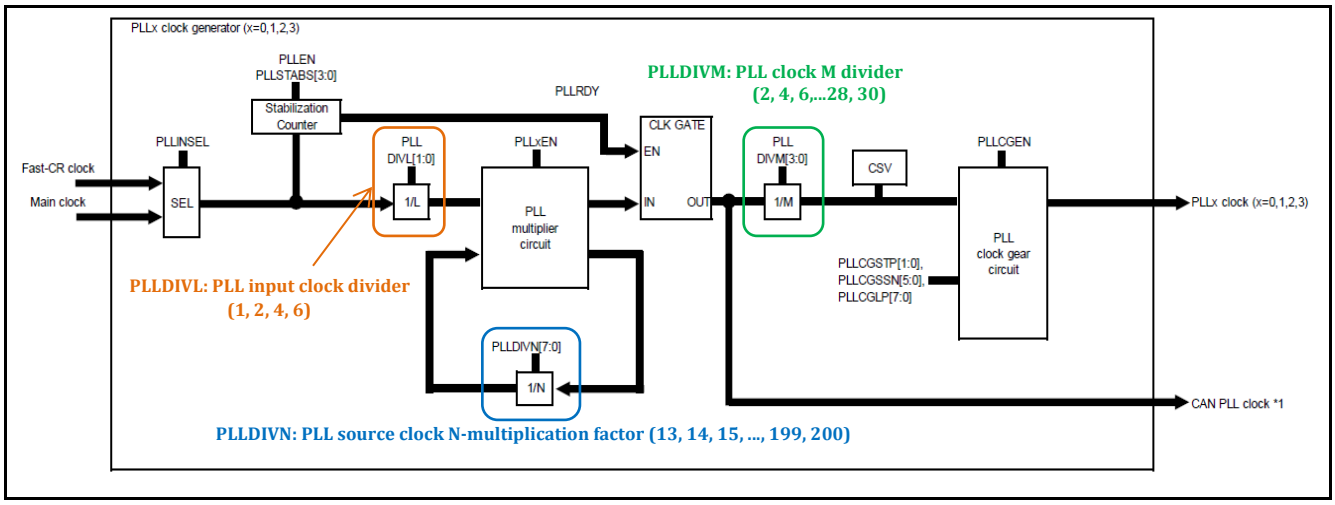


Table 1. PLL Clock Configuration Example

Input Clock	Input Clock Divider Configuration (Set by PLLDIVL)	PLLin	PLL Multiplier Configuration (Set by PLLDIVN)	PLLout	Output Divider Configuration (Set by PLLDIVM)	PLL Clock
4 MHz	1	4 MHz	120	480 MHz	2	240 MHz
4 MHz	1	4 MHz	144	576 MHz	4	144 MHz
4 MHz	1	4 MHz	120	480 MHz	4	120 MHz
4 MHz	1	4 MHz	144	576 MHz	6	96 MHz
4 MHz	1	4 MHz	120	480 MHz	6	80 MHz

Calculate the PLL clock frequency as follows:

PLLin = (input clock) / (divider setting by PLLDIVL)

PLLout = PLLin × (multiplier setting by PLLDIVN)

PLL clock = PLLout / (divider setting by PLLDIVM)

The permission range of the PLL output clock (PLLout) is different for each product. Set the multiplication value so that the PLL output clock (PLLout) is within the following:

- S6J3118, S6J3119, S6J311A, S6J311B, S6J311C, S6J311D, S6J311E: 400 MHz to 576 MHz
- S6J3128, S6J3129, S6J312A: 400 MHz to 512 MHz
- S6J3200 series: 400 MHz to product specification [2]
- S6J3300, S6J3350 series: 200 MHz to product specification [2]
- S6J3360, S6J3370, S6J3510 series: 200 MHz to product specification [2]
- S6J3400 series: 200 MHz to 320 MHz

[2] Refer to the datasheet. Maximum value depends on function digit and PLL/SSCG clock number and PLL type.

For the maximum operation frequency of the CPU clock, see the datasheet of the target products.

- S6J311B, S6J311C, S6J311D, S6J311E: 144 MHz
- S6J3118, S6J3119, S6J311A: 96 MHz
- S6J3128, S6J3129, S6J312A: 128 MHz
- S6J32xC: 240 MHz (x = 3, 4, 5, 6, 7, 8),
- S6J32xA: 160 MHz (x = A, B, C, D)
- S6J32xE: 240 MHz (x = E, F, G, K, L, M, N)
- S6J3300, S6J3350 series: 240 MHz
- S6J3360, S6J3370, S6J3400, S6J3510 series: 132 MHz

For more information, refer to the datasheet and hardware manual for each product.

2.3 Clock Distributor

The clock distributor has a source clock selector and a clock divider for each clock domain. The Traveo family microcontrollers have the following clock domains, and only one source clock is selected for each clock domain. Each clock domain uses the selected source clock. The source clock is divided to become the divided clock, which is distributed as an internal operating clock.

- Clock Domain 0 (CD0)
- Clock Domain 1/2/3/4/5 (CD1/CD2/CD3/CD/4/CD5)
- TRC Clock Domain (CD_TRC)
- HSSPI Clock Domain (CD_HSSPI)
- MCUC Clock Domain (CD_MCUC)
- CLKO Clock Domain (CD_CLKO)

The configuration of the clock domain is different for each product. Therefore, refer to the appropriate hardware manual.

Figure 4 and Figure 5 show the clock distributor block of Clock Domain 0 and the MCUC Clock Domain of the S6J3110 series as an example. After the source clock is selected, the divided clock of the source clock is distributed as an internal operating clock. For details on the clock domain block and the maximum frequency of the internal clock, refer to the datasheet and hardware manual of each product.

Figure 4. Clock Distributor Block Diagram (Clock Domain 0)

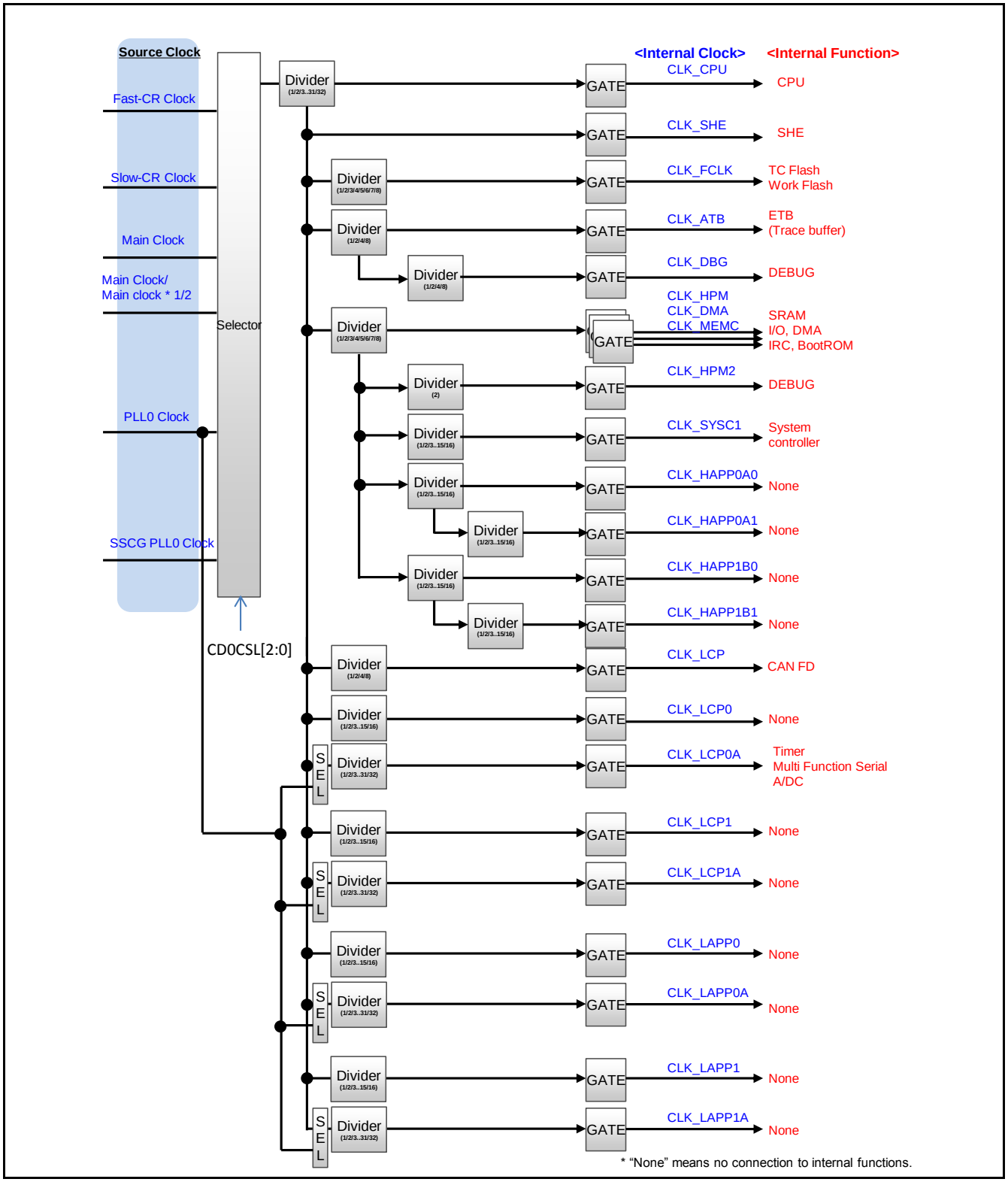
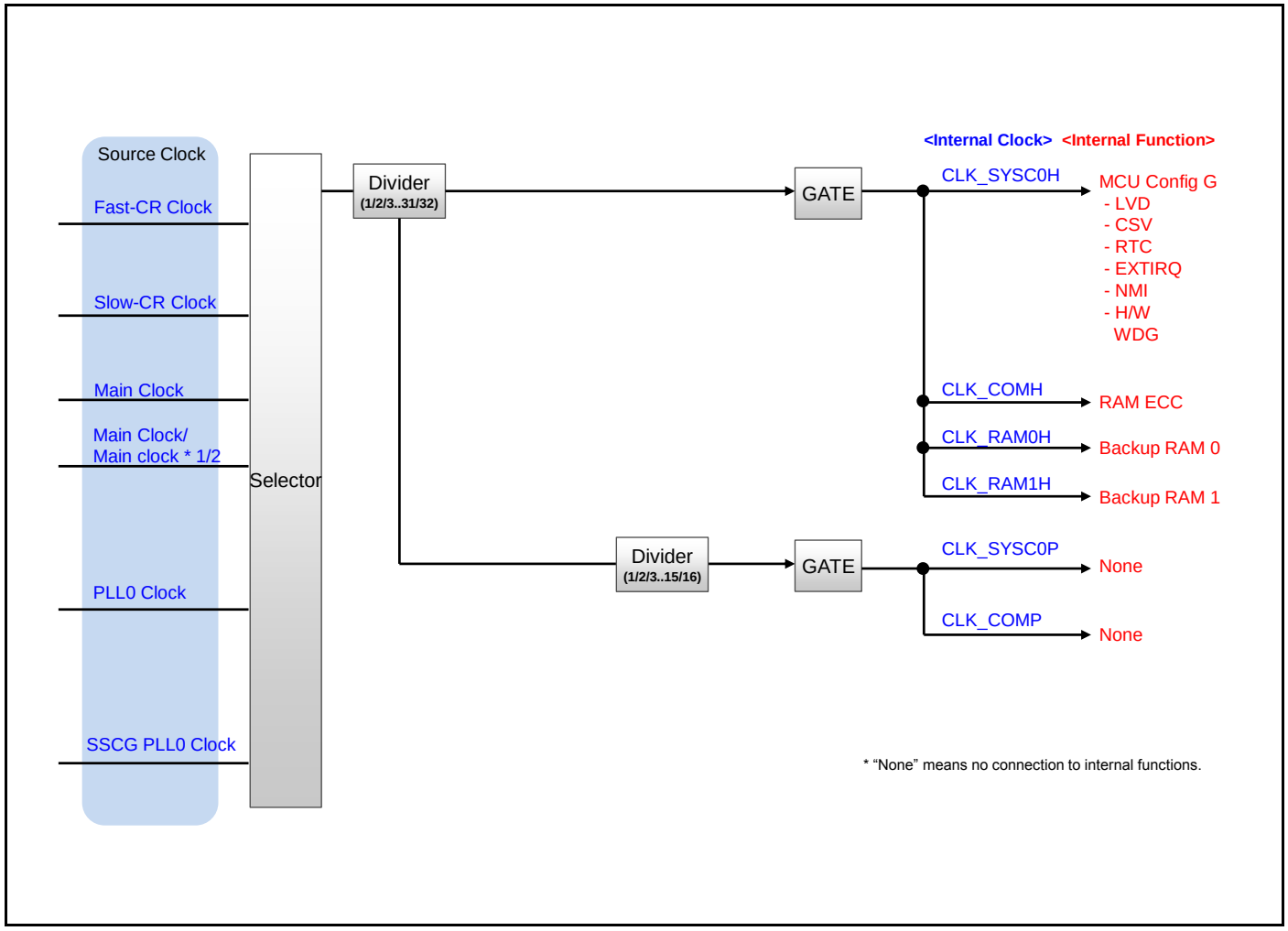


Figure 5. Clock Distributor Block Diagram (MCUC Clock Domain)



2.4 How to Confirm Combination of Peripheral Function and Its Source Clock

If you want to know a combination of a peripheral function and its source clock for the S6J3200/ 3300/ 3350/ 3360/ 3370/ 3400/ 3510 series, you can confirm it using the following process.

1. See the group name from the table base address map in the S6J3xxx Series Hardware Manual. For the S6J3400 series, the Base Timer function belongs to the Common PERI #1 group, as shown in Figure 6.

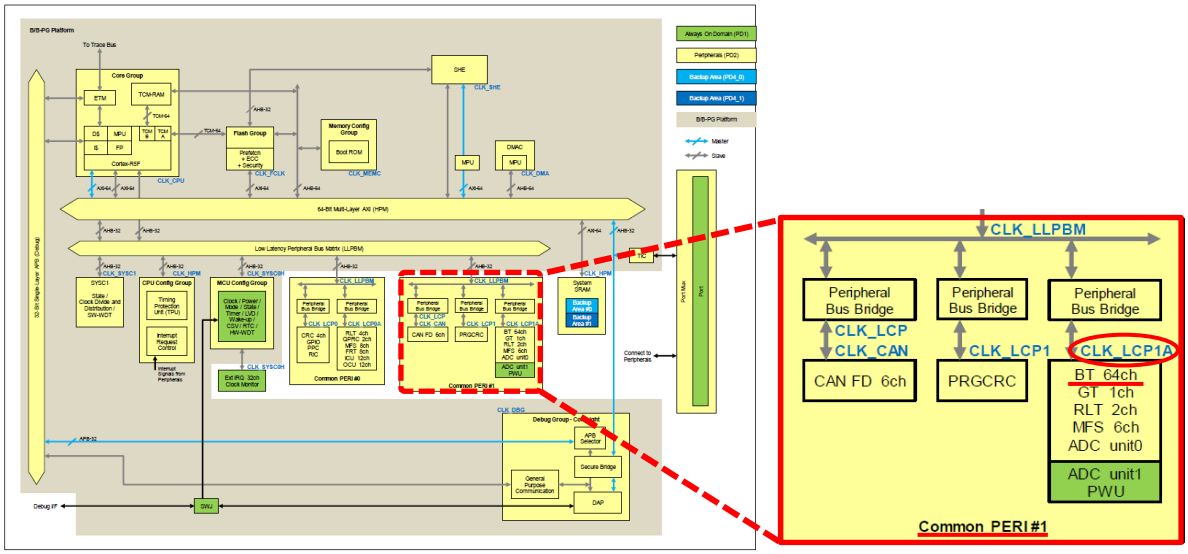
Figure 6. Base Address Map in S6J3400 Series Hardware Manual

CHAPTER 7: Memory and Base Address Map

START Address	END Address	Group	Function	PPU_No
B484_D000	B484_D3FF	Common PERI #1	Base Timer ch.52	318
B484_D400	B484_D7FF	Common PERI #1	Base Timer ch.53	319
B484_D800	B484_DBFF	Common PERI #1	Base Timer ch.54	320
B484_DC00	B484_DFFF	Common PERI #1	Base Timer ch.55	321
B484_E000	B484_E3FF	Common PERI #1	Base Timer ch.56	322
B484_E400	B484_E7FF	Common PERI #1	Base Timer ch.57	323
B484_E800	B484_EBFF	Common PERI #1	Base Timer ch.58	324
B484_EC00	B484_EFFF	Common PERI #1	Base Timer ch.59	325

2. The group name and its clock source are described in “PLATFORM OVERVIEW Configuration” and “CLOCK SYSTEM” in the Traveo Platform Family Hardware Manual and “Block Diagram” in the S6J3xxx Series Hardware Manual. For the S6J3400 series, you can confirm that 64 channels of Base Timer (BT) belong to Common PERI #1, and its clock source is CLK_LCP1A, as shown in Figure 7.

Figure 7. Block Diagram of S6J3400 Series



3. For the internal clock frequency, refer to the datasheet of each product. The clock frequency of CLK_LCP1A, which is the clock source of Base Timer, is up to 40 MHz, as shown in Figure 8.

Figure 8. Internal Clock Frequency of S6J3400 Series

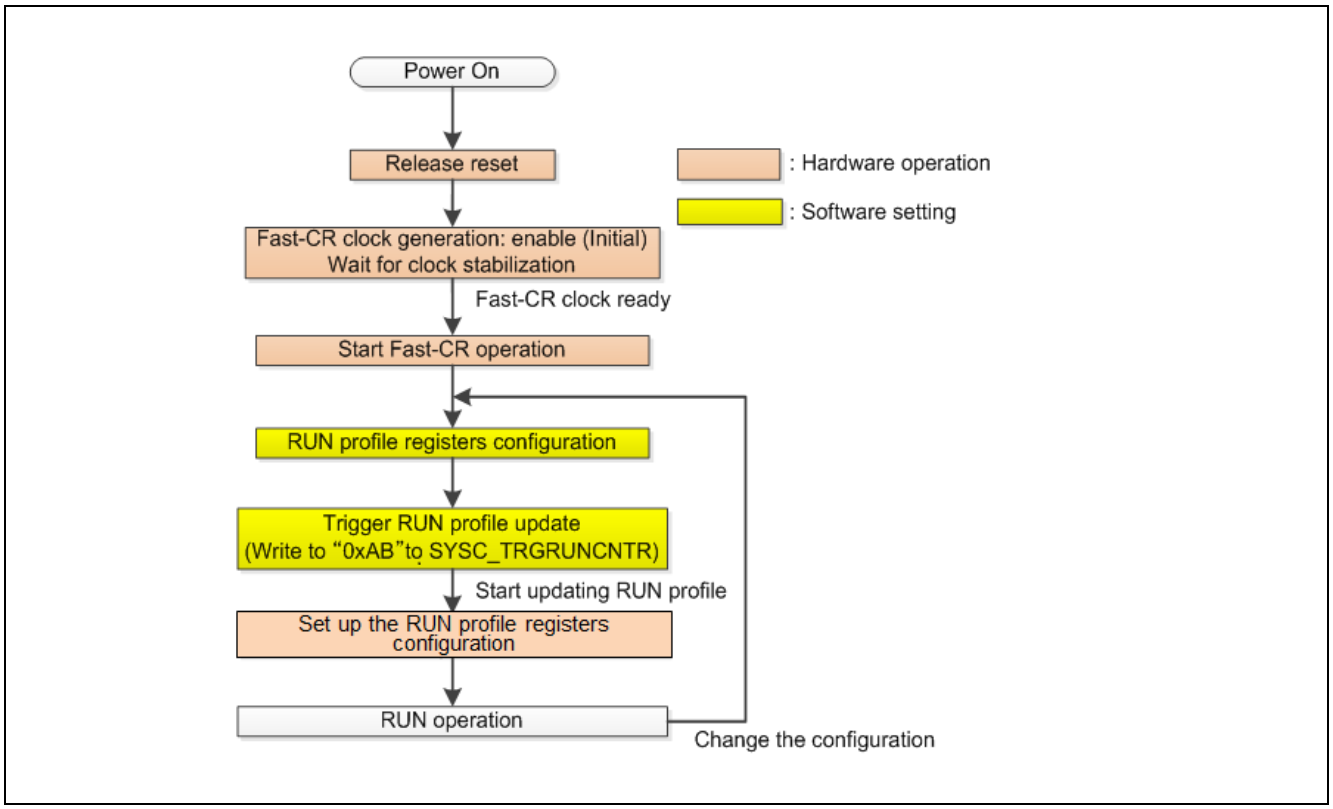
Parameter	Symbol	Pin Name	Conditions	Value				Unit	Remarks
				Min	Typ	Max *1	Max *2		
Internal clock frequency	$f_{SSCG0out}$	-	-	200	-	320	320	MHz	
	$f_{PLL0out}$	-	-	200	-	320	320	MHz	
	f_{CLK_CPU}	-	-	-	-	132	80	MHz	
	f_{CLK_SHE}	-	-	-	-	33	40	MHz	
	f_{CLK_FCLK}	-	-	-	-	66	80	MHz	
	f_{CLK_ATB}	-	-	-	-	66	40	MHz	
	f_{CLK_DBG}	-	-	-	-	66	40	MHz	
	f_{CLK_HPM}	-	-	-	-	33	40	MHz	
	f_{CLK_DMA}	-	-	-	-	33	40	MHz	
	f_{CLK_MEMC}	-	-	-	-	33	40	MHz	
	f_{CLK_SYSC1}	-	-	-	-	33	40	MHz	
	f_{CLK_LLPBM}	-	-	-	-	132	80	MHz	
	f_{CLK_LCP}	-	-	-	-	66	80	MHz	
	f_{CLK_LCP0}	-	-	-	-	33	40	MHz	
	f_{CLK_LCP0A}	-	-	-	-	40	40	MHz	
	f_{CLK_LCP1}	-	-	-	-	33	40	MHz	
	f_{CLK_LCP1A}	-	-	-	-	40	40	MHz	

3 Clock Setting Procedure

3.1 Clock Setting Procedure for RUN Configuration

Figure 9 depicts an example of the clock setting procedure for the RUN configuration.

Figure 9. Clock Setting Procedure Example for RUN Configuration



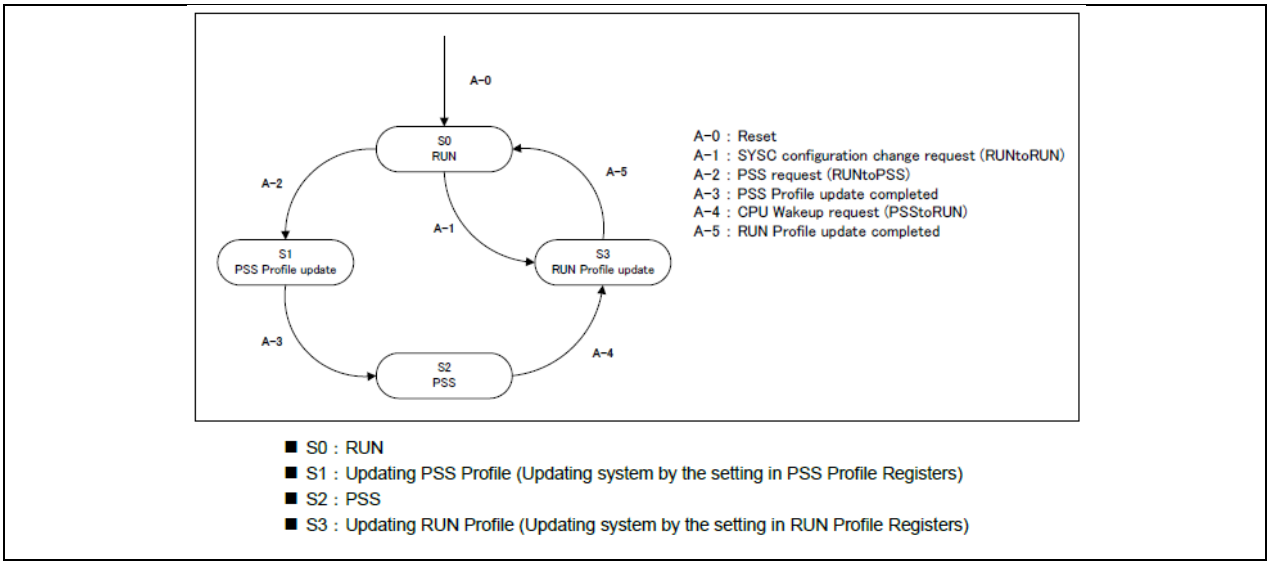
The microcontroller operates with the fast-CR clock after power on. When changing the system clock frequency, it is necessary to set the RUN profile with software.

3.1.1 RUN Profile Configuration

This product supports two state operations, as shown in Figure 10, and has a wide variety of power consumption settings.

- **RUN (normal operation):** This is the state in which the CPU operates to run programs. The CPU operates in this state after an initialization reset. In this state, RUN profile update and transition to PSS are possible.
- **PSS (power-saving state):** This is the state in which the CPU has stopped running programs and can be set to low-power consumption.

Figure 10. State Transition Diagram



For each RUN/PSS state, the following items related to the clock system can be configured via the RUN profile register or the PSS profile register. If you change the clock setting, you must update the profile.

- Source clock oscillation enable/stop
- Clock domain control (source selection, division, oscillation enable, and stop of each domain clock)

3.1.2 RUN Profile Settings Update

When changing the RUN/PSS profile setting parameters, the RUN profile settings are not updated by only writing to the register. It is necessary to do the following to update the settings of the RUN profile.

1. Write the settings to the register.
2. Clear the RUN profile update completion flag at the system status flag interrupt clear register (SYSC0_SYSICLR).
3. Write '0xAB' to the RUN profile update trigger register of the system controller (SYSC0_TRGRUNCNTR).
4. Poll the RUN profile update completion flag bit of the system status register (SYSC0_SYSSTSR: RUNDF0 bit), and wait until the completion of the profile update (RUNDF0 = '1').

If the contents of the new RUN profile do not have any error conditions, the control circuit will update the contents of the profile as follows:

- '1' is set in the system status register (SYSC0_SYSSTSR: RUNSTS0).
- The contents of the RUN profile are copied to the APPLIED profile.
- Settings are updated in the following order:
  - Clock oscillation enable/stop (including waiting for oscillation stabilization)
  - Clock supervisor setting changes
  - LVD setting changes
  - Clock operation settings (source clock changes, division, and ON/OFF of each clock source)
  - Clock stop settings (source clock stop)
- When the RUN profile update is completed, the system status register (SYSC0_SYSSTSR: RUNSTS0) is cleared to '0'.
- '1' is set in the system status register (SYSC0_SYSSTSR: RUNDF0).

If the present settings have error conditions, a profile error occurs, and system error interrupt factor register 1 (SYSC0_SYSERRIR1: RUNERRIF0) is set to '1'. In this case, the contents of the new profile are discarded, and the circuit operates with the contents of the profile currently in use.

Notes:

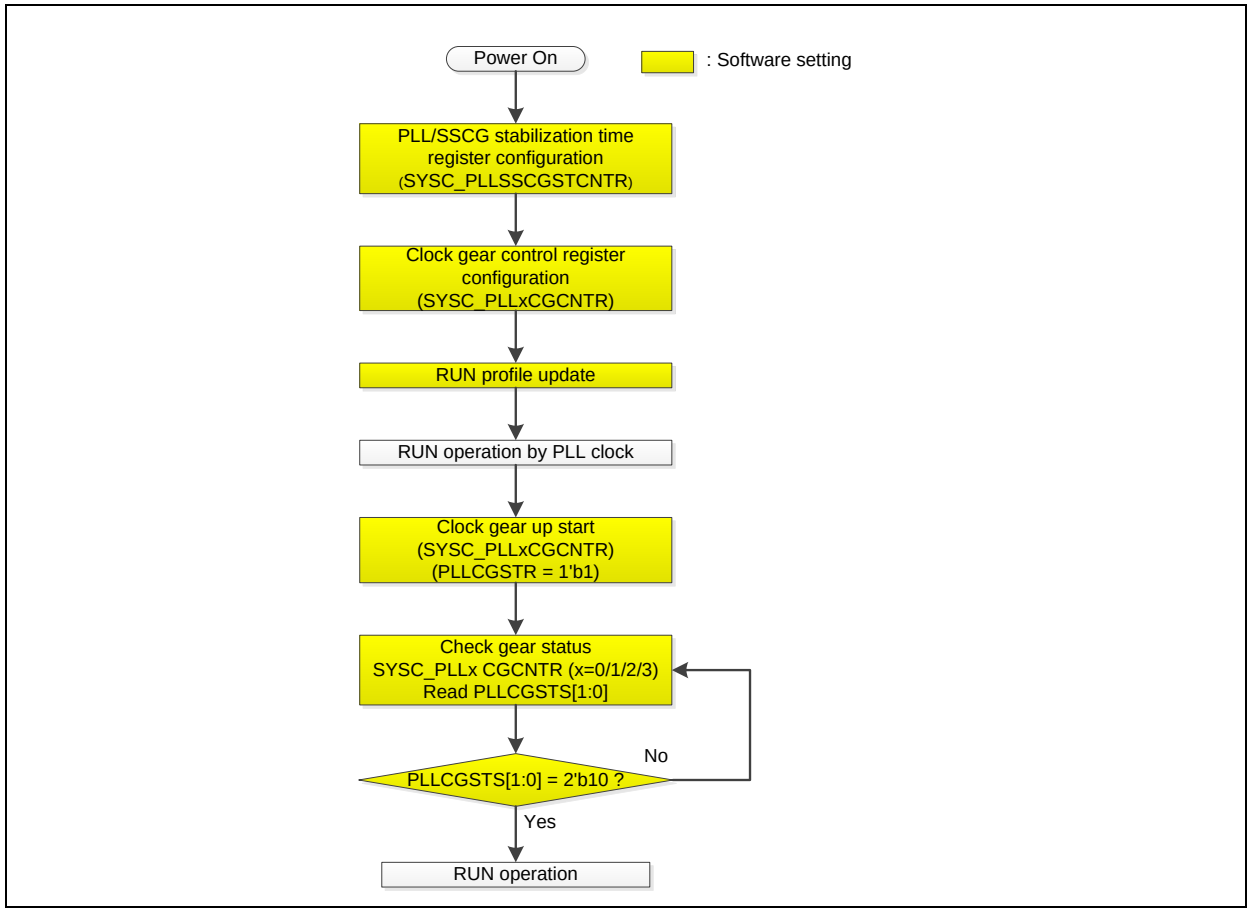
1. Profile update is prohibited during a profile update. If the profile is updated again while the profile is being updated, a system error interrupt (SYSC0_SYSERRIR1: RUNTRGERRIF) will be generated, and the attempted update profile will be disabled.
2. Before updating the RUN profile, it is necessary to confirm that the flag of the profile status register (SYSC0_SYSPROSTSR:RUNPSTS) is already cleared. When the RUN profile is updated during a profile error status, a nonmaskable interrupt (NMI) will be generated, and the RUN profile settings will be discarded.
3. During a RUN profile update, write access to the RUN profile register group is prohibited. If a write access is made to the RUN profile register group during a RUN profile update, the write data will be disabled, and a bus error will occur.

For more information about profile settings and operation, refer to the chapter "Low-Power Consumption" in the hardware manual.

3.2 Clock Setting Procedure for PLL Configuration

Figure 11 shows the setting procedure example for using the PLL clock. After updating the RUN profile settings, the frequency of the system clock changes gradually via operation of the clock gear.

Figure 11. Clock Setting Procedure Example for Using PLL Clock



3.2.1 Enable Setting of Source Clock

After the hardware reset, source clocks from the external/built-in oscillation circuits (fast-CR circuit clock, slow-CR clock, and main clock/main clock divided by 2) enter the oscillation-enabled state. Conversely, the PLL clock enters the oscillation-disabled state. It is necessary to enable oscillation of the PLL clock when using PLL and changing the source clock of the system clock to the PLL clock. To enable PLL oscillation, the oscillation for the main clock must already be enabled. If the oscillation for the main clock has not been enabled, a profile error occurs.

To enable the oscillation of the corresponding clock during the RUN state, write '1' to the bits of the register described in Table 2. However, control for enabling/disabling the source clock oscillation is available only when the system is not using the source clock. Fast-CR clock/slow-CR clock oscillation cannot be disabled in the RUN state.

Table 2. Source Clock Oscillation Enabled Setting Example

Register Abbreviation	Bit Name	Settings	Value (Decimal)
SYSC0_RUNCKSRER	PLLxEN	Enable PLLx clock oscillation	1
	MOSCEN	Enable main clock oscillation	1
	SCROSCEN	Enable slow-CR clock oscillation	1
	CROSCEN	Enable fast-CR clock oscillation	1

3.2.2 Source Clock Timer Setting

The source clock timer is a timer for gating the clock output until the end of the clock oscillation stabilization wait time. To use the source clock timer, it is necessary to set the registers as described in Table 3.

Table 3. Source Clock Timer Setting

Register Abbreviation	Bit Name	Settings
SYSC_MOCTCPR	PSCL	Select the division ratio of the input clock of the main clock timer.
	CMPR	Compare the value of the main clock timer.
SYSC_ MOCTTRGR	CGCPT	Change the timer settings/start the timer counting.
SYSC_PLLSSCGSTCNTR	PLLSTABS	PLL stabilization wait time selection

3.2.3 Clock Gear

The clock gear outputs the gear clock, which is gradually changed by the step-by-step outputting of the input clock, so it is possible to shift the operating frequency gradually. The frequency fluctuates abruptly when switching from the main clock to the PLL clock, so the power supply current also fluctuates considerably. A clock gear operation is necessary to prevent overshoot/undershoot of the power supply current at the clock switching time.

To use the clock gear, it is necessary to set the register as described in Table 4.

Table 4. Clock Gear Setting

Register Abbreviation	Bit Name	Settings
SYSC_PLL0CGCNTR	PLLCGLP [3]	Loop count per one step of the PLL clock gear operation
	PLLCGSTP [3]	Step width at the PLL clock gear-up/down
	PLLCGSSN [3]	Step at the start of the PLL clock gear operation
	PLLCGSTR	Start gear operation
	PLLCGEN [3]	Clock gear operation enabled

[3] Do not change the setting after PLL clock oscillation is set to 'enable'.

3.2.4 Clock Gear Operation Start Procedure

The clock gear starts the operation by performing the following steps after setting the register described in Table 4 (except PLLCGSTR). It is necessary to wait until the clock gear is completed after the clock gear operation begins.

1. Select the PLL clock as the domain clock.
2. Set '01' to PLLCGSTR of the PLL clock gear control register (SYSC_PLL0CGCNTR).
3. Poll the value of the clock gear status flag (SYSC_PLL0CGCNTR.PLLCGSTS), and wait until the clock indicates the stop state (PLLCGSTS = '10').

Notes:

1. The smaller the step width and the larger the loops count, the more gradually the frequency changes.
2. When clock gear operation is enabled, after the oscillation stabilization wait time of the clock gear input clock, the clock is output according to the setting of PLLCGSSN. However, it does not gear up/gear down until the clock gear operation start setting is performed.

3.2.5 PLL Clock Multiplier and Divider

A PLL clock multiplier and divider can be set via the RUN PLLx control register (SYSC0_RUNPLLxCNTR). For an example of PLL clock configuration, see Table 5.

Table 5. Clock Multiplier and Divider Setting Example

Register Abbreviation	Bit Name	Settings
SYSC0_RUNPLLxCNTR	PLLDIVN	PLL clock N-multiplier setting
	PLLDIVM	PLL clock M-divider setting
	PLLDIVL	PLL input clock divider setting

3.2.6 Source Clock for Clock Domain 0

A hardware reset results in the fast-CR clock being selected as the clock used in all clock domains. It is necessary to set the PLL clock to the source clock of clock domain 0 when changing the source clock of the system clock to the PLL clock. The source clock of the system clock becomes the PLL clock by setting the source clock of clock domain 0 to PLL clock.

The source clock of clock domain 0 during the RUN state can be selected by setting the register in Table 6. The example describes setting the source clock of clock domain 0 to PLL clock.

Table 6. Clock Domain 0 Source Clock Setting Example

Register Abbreviation	Bit Name	Settings	Value (decimal)
SYSC1_ RUNCKSELR0	CD0CSL	Clock domain 0 source clock selection	4

3.3 Clock Setting Procedure of Sample Software

The clock setting procedure of the sample software is shown in Figure 12.

Figure 12. Clock Setting Example Using PLL in Sample Software (*start.c*)

```
static void ConfigureClocks(void)
{
   // At first, disable PSS profile update.
   // This setting is for wakeup from shutdown mode.
   // note: SYSC1 was cleared by hardware after PSS profile is updated, but SYSC0 was not cleared.
   SYSC0_PROTKEYR     = SYSC_KEY_UNLOCK;
   SYSC0_PSSENR_PSSEN0 = 0;

   Stabilization time setting with Source Clock Timer

//  Set main oscillator and main PLL stabilization time        Stabilization time setting for Main clock and PLL/SSCG clock.
                                    //-------------------------------------------------------------------

      // Set new main oscillation stabilization time and trigger data update
         SYSC0_PROTKEYR              =  SYSC_KEY_UNLOCK;   // unlock SYSC0
         SYSC_6.unMOCTCPR.stcField    =  (stc_sysc_6_moctcpr_field_t){  .u4PSCL  = SYSC_MAINSCT_PRESCALER,  // pre-scaler
                                          .u16CMPR  = SYSC_MAINSCT_CMPR };     // compare value
         // trigger configuration capture
         SYSC0_PROTKEYR              =  SYSC_KEY_UNLOCK;   // unlock SYSC0
         SYSC_6.unMOCTTRGR.stcField    =  (stc_sysc_6_mocttrgr_field_t){  .u1CGCPT    = 1 };

      // Set PLL / SSCG stabilization time
         SYSC0_PROTKEYR              =  SYSC_KEY_UNLOCK;   // unlock SYSC0
         SYSC_7.unPLLSSCGSTCNTR.stcField = (stc_sysc_7_pllsscgstcntr_field_t){  .u4PLLSTABS  = SYSC_PLLST_PLLSTABS,
                                                .u4SSCGSTABS = SYSC_PLLST_PLLSTABS, };

   RUN Profile configuration (Power Domain setting)

         //  Configure run profile
         //-------------------------------------------------------------------
      // Enable power domains
         SYSC0_PROTKEYR              =  SYSC_KEY_UNLOCK;   // unlock SYSC0
         SYSC0_1.unRUNPDCFGR.stcField   =  (stc_sysc0_1_runpdcfgr_field_t){   .u1PD6_1EN  =  1,  // switch on PD6_1 ()
                                                .u1PD6_0EN  =  1,  // switch on PD6_0 ()
                                                .u1PD5_3EN  =  1,  // switch on PD5_3 ()
                                                .u1PD5_2EN  =  1,  // switch on PD5_2 ()
                                                .u1PD5_1EN  =  1,  // switch on PD5_1 ()
                                                .u1PD5_0EN  =  1,  // switch on PD5_0 ()
                                                .u1PD4_1EN  =  1,  // switch on PD4_1 (Backup RAM1)
                                                .u1PD4_0EN  =  1,  // switch on PD4_0 (Backup RAM0)
                                                .u1PD3EN    =  1,  // always on (Core)
                                                .u1PD2EN    =  1 };// always on (Peripheral)
```

- Input protect key of SYSC
- Set the Prescaler and compare value

- Input protect key of SYSC
- Change the main clock timer setting and start the timer counting

- Input protect key of SYSC
- Set the PLL and SSCG stabilization time

- Input protect key of SYSC
- Power Domain enable setting.

RUN Profile configuration (Enable Oscillator)

```
// Enable oscillators
  SYSC0_PROTKEYR                    =   SYSC_KEY_UNLOCK;     // unlock SYSC0
  SYSC0_1.unRUNCKSRER.stcField      =   (stc_sysc0_1_runcksrer_field_t){    .u1SOSCEN    =   0,   // disable Sub Oscillation
                                                                            .u1PLL0EN    =   1,   // enable PLL0
                                                                            .u1SSCG0EN   =   1,   // enable SSCG0
                                                                            .u1MOSCEN    =   1,   // enable Main Oscillation
                                                                            .u1CROSCEN   =   1,   // RC Osc. (always on in RUN state)
                                                                            .u1SCROSCEN =    1 };// Slow RC Osc. (always on in RUN state)
```

- Input protect key of SYSC
- Enable Oscillator setting (Except for sub oscillation)

RUN Profile configuration (PLL/SSCG setting)

```
//-----------------------------------------------------------------------
  // Write Main / SSCG PLL settings
  SYSC0_PROTKEYR                    =   SYSC_KEY_UNLOCK;     // unlock SYSC0
  SYSC0_1.unRUNPLL0CNTR.stcField    =   (stc_sysc0_1_runpll0cntr_field_t){  .u8PLL0DIVN =   (SYSC_MAIN_PLL_DIVN),   // set PLL input multiplication value
                                                                            .u4PLL0DIVM =   (SYSC_MAIN_PLL_DIVM),   // set PLL output divider
                                                                            .u2PLL0DIVL =   (SYSC_PLL_DIVL)};       // set PLL input divider

  SYSC0_PROTKEYR                    =   SYSC_KEY_UNLOCK;     // unlock SYSC0
  SYSC0_1.unRUNSSCG0CNTR0.stcField = (stc_sysc0_1_runsscg0cntr0_field_t){ .u8SSCG0DIVN =   (SYSC_MAIN_SSCG_DIVN),   // set SSCG PLL input multiplication value
                                                                            .u4SSCG0DIVM =   (SYSC_MAIN_SSCG_DIVM),   // set SSCG PLL output divider
                                                                            .u2SSCG0DIVL =   (SYSC_SSCG_DIVL)};       // set SSCG PLL input divider
```

RUN Profile configuration (Clock gear)

```
// Clock gear
  SYSC0_PROTKEYR                    =   SYSC_KEY_UNLOCK;     // unlock SYSC0
  SYSC_7.unPLL0CGCNTR.stcField      =   (stc_sysc_7_pll0cgcntr_field_t){    .u8PLLCGLP   =   4,   // Loops per step
                                                                            .u2PLLCGSTP =    1,   // 2 steps
                                                                            .u6PLLCGSSN =    8,   // Start step = 8
                                                                            .u1PLLCGSTR =    0,   // Start gear operation
                                                                            .u1PLLCGEN   =   1}; // Enable
```

- Input protect key of SYSC
- PLL clock gear setting.

```
  SYSC0_PROTKEYR                    =   SYSC_KEY_UNLOCK;     // unlock SYSC0
  SYSC_7.unSSCG0CGCNTR.stcField     =   (stc_sysc_7_sscg0cgcntr_field_t){   .u8SSCGCGLP   =   4,   // Loops per step
                                                                            .u2SSCGCGSTP =   1,   // 2 steps
                                                                            .u6SSCGCGSSN =   8,   // Start step = 8
                                                                            .u1SSCGCGSTR =   0,   // Start gear operation
                                                                            .u1SSCGCGEN   =  1}; // Enable
```

- Input protect key of SYSC
- SSCG clock gear setting.

RUN Profile configuration (Clock source selection)

```
//-----------------------------------------------------------------------
  // Select clock sources
  SYSC0_PROTKEYR                    =   SYSC_KEY_UNLOCK;     // unlock SYSC0
  SYSC0_1.unRUNCKSELR.stcField      =   (stc_sysc0_1_runckselr_field_t){   .u3CDMCUCCSL=   5}; // Clock domain MCUC clock = SSCG0

  SYSC1_PROTKEYR                    =   SYSC_KEY_UNLOCK;     // unlock SYSC1
  SYSC1.unRUNCKSELR0.stcField       =   (stc_sysc1_runckselr0_field_t){     .u4HSSPICSL =    0,   // Hsspi clock domain = Fast CR
                                                                            .u1LAPP1ACSL=    0,   // LAPP1A clock = CD0 (1=PLL0)
                                                                            .u1LAPP0ACSL=    0,   // LAPP0A clock = CD0 (1=PLL0)
                                                                            .u1LCP1ACSL =    0,   // LCP1A clock = CD0 (1=PLL0)
                                                                            .u1LCP0ACSL =    0,   // LCP0A clock = CD0 (1=PLL0)
                                                                            .u3CD0CSL    =   5}; // Clock Domain 0 = SSCG0
```

- Input protect key of SYSC
- Clock domain MCUC clock
  → Select SSCG
- Input protect key of SYSC
- Clock domain 0 clock
  → Select SSCG
- LAPP1A/LAPP0A/LCP1A/LCP0A
  → Select CD0

```
SYSC1_PROTKEYR                  =   SYSC_KEY_UNLOCK;      // unlock SYSC1
SYSC1.unRUNCKSELR2.stcField     =   (stc_sysc1_runckselr2_field_t){     .u3TRCCSL   =   5}; // TRC clock = SSCG0
```

RUN Profile configuration (Enable clocks)

```
//------------------------------------------------------------------
 // Enable clocks
 SYSC1_PROTKEYR                  =   SYSC_KEY_UNLOCK;      // unlock SYSC1
 SYSC1.unRUNCKER1.stcField       =   (stc_sysc1_runcker1_field_t) {      .u1ENCLKCD3B1   =   0,   // Disable CD3B1
                                                                         .u1ENCLKCD3B0   =   0,   // Disable CD3B0
                                                                         .u1ENCLKCD3A1   =   0,   // Disable CD3A1
                                                                         .u1ENCLKCD3A0   =   0,   // Disable CD3A0
                                                                         .u1ENCLKCD3     =   0,   // Disable CD3
                                                                         .u1ENCLKCD2B1   =   0,   // Disable CD2B1
                                                                         .u1ENCLKCD2B0   =   0,   // Disable CD2B0
                                                                         .u1ENCLKCD2A1   =   0,   // Disable CD2A1
                                                                         .u1ENCLKCD2A0   =   0,   // Disable CD2A0
                                                                         .u1ENCLKCD2     =   0,   // Disable CD2
                                                                         .u1ENCLKCD1B1   =   0,   // Disable CD1B1
                                                                         .u1ENCLKCD1B0   =   0,   // Disable CD1B0
                                                                         .u1ENCLKCD1A1   =   0,   // Disable CD1A1
                                                                         .u1ENCLKCD1A0   =   0,   // Disable CD1A0
                                                                         .u1ENCLKCD1     =   0,   // Disable CD1
                                                                         .u1ENCLKHSSPI   =   0}; // Disable HSSPI
 SYSC1_PROTKEYR                  =   SYSC_KEY_UNLOCK;      // unlock SYSC1
 SYSC1.unRUNCKER2.stcField       =   (stc_sysc1_runcker2_field_t) {      .u1ENCLKCD5B1   =   0,   // Disable CD5B1
                                                                         .u1ENCLKCD5B0   =   0,   // Disable CD5B0
                                                                         .u1ENCLKCD5A1   =   0,   // Disable CD5A1
                                                                         .u1ENCLKCD5A0   =   0,   // Disable CD5A0
                                                                         .u1ENCLKCD5     =   0,   // Disable CD5
                                                                         .u1ENCLKCD4B1   =   0,   // Disable CD4B1
                                                                         .u1ENCLKCD4B0   =   0,   // Disable CD4B0
                                                                         .u1ENCLKCD4A1   =   0,   // Disable CD4A1
                                                                         .u1ENCLKCD4A0   =   0,   // Disable CD4A0
                                                                         .u1ENCLKCD4     =   0}; // Disable CD4
```

- Input protect key of SYSC
- Disable clocks

- Input protect key of SYSC
- Disable clocks

RUN Profile configuration (Clock divider setting)

```
//------------------------------------------------------------------
// Set clock dividers (valid setting for all main-PLL frequencies)
SYSC0_PROTKEYR                  =   SYSC_KEY_UNLOCK;      // unlock SYSC0
SYSC0_1.unRUNCKDIVR.stcField    =   (stc_sysc0_1_runckdivr_field_t) {   .u4MCUCPDIV   =   0,   // MCUconfig APB clock divider = 1
                                                                        .u5MCUCHDIV   =   3 };// MCUconfig AHB clock divider = 4
// Set clock dividers (valid setting for all main-PLL frequencies)
SYSC1_PROTKEYR                  =   SYSC_KEY_UNLOCK;      // unlock SYSC1
SYSC1.unRUNCKDIVR0.stcField     =   (stc_sysc1_runckdivr0_field_t) {    .u3HPMDIV     =   3,   // HPM clock divider = 4
                                                                        .u5TRCDIV     =   3,   // TRC Clock divider = 4
                                                                        .u2DBGDIV     =   0,   // DBG Clock divider = 1
                                                                        .u2ATBDIV     =   1,   // ATB Clock divider = 2
                                                                        .u5SYSDIV     =   0 };// System Clock divider = 1
```

- Input protect key of SYSC
- Divider setting for Clock domain MCUC block

- Input protect key of SYSC
- Divider setting for following clock
  - -HPM
  - -TRC
  - -DBG
  - -ATB
  - -System clock

```
SYSC1_PROTKEYR                  =   SYSC_KEY_UNLOCK;     // unlock SYSC1
SYSC1.unRUNCKDIVR1.stcField     =   (stc_sysc1_runckdivr1_field_t) {   .u4HAPP1B1DIV   =   0,   // HAPP1B1 clock divider = 1
                                                                       .u4HAPP1B0DIV   =   0,   // HAPP1B0 Clock divider = 1
                                                                       .u4HAPP0A1DIV   =   0,   // HAPP0A1 Clock divider = 1
                                                                       .u4HAPP0A0DIV   =   0,   // HAPP0A0 Clock divider = 1
                                                                       .u4SYSC1DIV     =   0,   // SYSC1 Clock divider = 1
                                                                       .u3EXTBUSDIV    =   7 };// EXTBUS Clock divider = 128
```

- Input protect key of SYSC
- Divider setting for following clock
  - -HAPP1B1DIV / HAPP1B0DIV
  - -HAPP0A1DIV / HAPP0A0DIV
  - -SYSC1
  - -External Bus

```
SYSC1_PROTKEYR                  =   SYSC_KEY_UNLOCK;     // unlock SYSC1
SYSC1.unRUNCKDIVR2.stcField     =   (stc_sysc1_runckdivr2_field_t) {   .u4LAPP1DIV     =   3,   // LAPP1 clock divider = 4
                                                                       .u4LAPP0DIV     =   3,   // LAPP0 Clock divider = 4
                                                                       .u4LCP1DIV      =   3,   // LCP1 Clock divider = 4
                                                                       .u4LCP0DIV      =   3,   // LCP0 Clock divider = 4
                                                                       .u2LCPDIV       =   1}; // LCP Clock divider = 2
```

- Input protect key of SYSC
- Divider setting for following clock
  - -LAPP1 / LAPP0
  - -LCP1 / LCP0
  - -LCP

```
SYSC1_PROTKEYR                  =   SYSC_KEY_UNLOCK;     // unlock SYSC1
SYSC1.unRUNCKDIVR3.stcField     =   (stc_sysc1_runckdivr3_field_t) {   .u5LAPP1ADIV    =   3,   // LAPP1A clock divider = 4
                                                                       .u5LAPP0ADIV    =   3,   // LAPP0A Clock divider = 4
                                                                       .u5LCP1ADIV     =   3,   // LCP1A Clock divider = 4
                                                                       .u5LCP0ADIV     =   3 };// LCP0A Clock divider = 4
```

- Input protect key of SYSC
- Divider setting for following clock
  - -LAPP1A / LAPP0A
  - -LCP1A / LCP0A

RUN Profile flag clear and start updating

```
//------------------------------------------------------------------
// Clear RUN Profile Done flag (SYSC_SYSSTSR_RUNDN)
SYSC0_PROTKEYR                  =   SYSC_KEY_UNLOCK;     // unlock SYSC0
SYSC0_SYSICLR_RUNDFCLR0         =   1;
```

- Input protect key of SYSC
- Clear the RUN profile update completion flag

```
// RUN Profile update enable
SYSC1_PROTKEYR                  =   SYSC_KEY_UNLOCK;     // unlock SYSC1
SYSC1_RUNENR_RUNEN1             =   SYSC_TRIGGER_APPLY_RUN_PROFILE;
```

- Input protect key of SYSC
- Setting RUN Profile enable

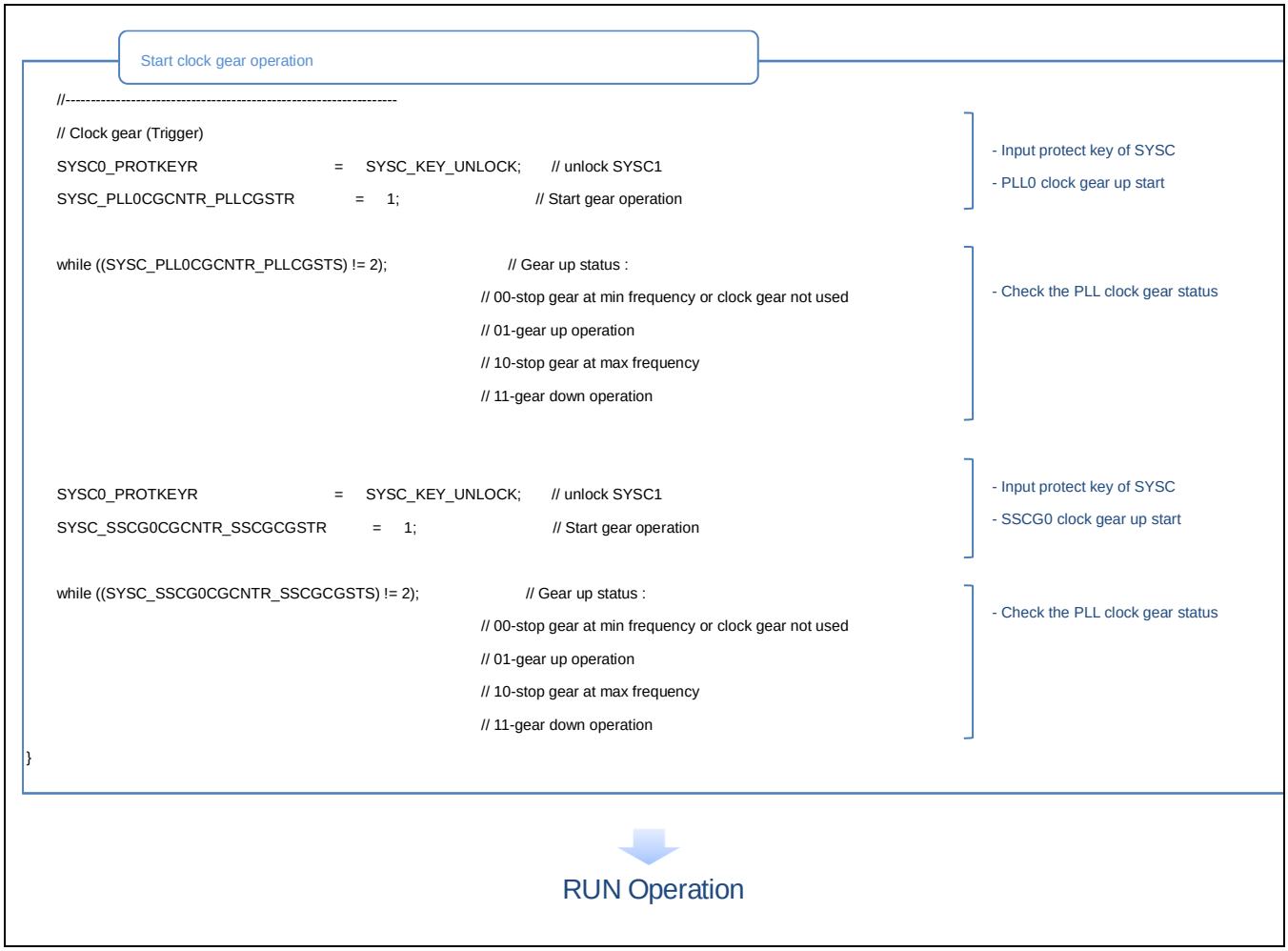
```
// Write the trigger value to apply the RUN profile
SYSC0_PROTKEYR                  =   SYSC_KEY_UNLOCK;     // unlock SYSC0
SYSC0_TRGRUNCNTR                =   SYSC_TRIGGER_APPLY_RUN_PROFILE;     // trigger RUN-->RUN transition
```

- Input protect key of SYSC
- Start updating the RUN Profile enables

RUN Profile updating check

```
// Wait until the RUN profile is applied
 while (SYSC0_SYSSTSR_RUNDF0 == 0);
```

- Wait until the RUN Profile is applied



4 Related Documents

4.1 Datasheets

- 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)
- S6J3200 Series Datasheet (Doc.No.002-05682)
- S6J32E/F/G Series Datasheet (Doc.No.002-10689)
- S6J3310/20/30/40 Series Datasheet (Doc.No.002-10635)
- S6J3350 Series Datasheet (Doc.No.002-10634)
- S6J3360/70 Series Datasheet (Doc.No.002-03359)
- S6J3400 Series Datasheet (Doc.No.001-97829)
- S6J3510 Series Datasheet (Doc.No.002-18647)
- MB9D560 Series Datasheet (Doc.No.002-05679)

4.2 Hardware Manuals

- S6J3120 Series Hardware Manual (Doc.No.002-04855)
- S6J3200 Series Hardware Manual (Doc.No.002-04852)
- S6J32E/F/G Series Hardware Manual (Doc.No.002-12500)
- Traveo Family Hardware Manual Platform Part for S6J3200 Series (Doc.No.002-04854)
- S6J3310/20/30/40/50 Series Hardware Manual (Doc.No.002-10185)
- Traveo Family HardwareManual Platform Part for S6J3310/3320/3330/3340/3350 Series (Doc.No.002-07884)
- 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 Hardware Manual (Doc.No.002-09919)
- Traveo Family Hardware Manual Platform Part for S6J3400 Series (Doc.No.002-07884)
- S6J3510 Series Hardware Manual (Doc.No.002-18642)
- Traveo Family Hardware Manual Platform Part for S6J3510 Series (Doc.No.002-07884)
- MB9D560 Series Hardware Manual (Doc.No.002-07882)

Document History

Document Title: AN204455 - How to Set Up the Clock System for the Traveo™ Family

Document Number: 002-04455

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	—	KHAS	07/31/2015	Initial release Added S6J3120 series to target products.
*A	5039879	KHAS	03/11/2016	Converted Spansion Application Note “S6J3110_AN708-00014” to Cypress format
*B	5269966	TANO	08/01/2016	Added S6J3200 series to target products. Updated template
*C	5405175	KOTH	09/15/2016	Added S6J3300/S6J3350/S6J3400 series to target products. Added how to confirm the combination of peripheral function and its source clock. Updated template
*D	5669326	KOTH	05/19/2017	Added S6J3360/S6J3370/S6J3510 series to target products.
*E	5868736	KOTH	09/21/2017	Added the description of S6J32K/L/M/N series

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

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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