Clock Configuration of TVII 1M Stater Kit

```
/************ Setting wait state for ROM **********/
CPUSS->unROM_CTL.stcField.u2SLOW_WS = 1u;
CPUSS->unROM_CTL.stcField.u2FAST_WS = 0u;
```

Bits	Name	SW	HW	Default or Enum	Description
0:1	SLOW_WS	RW	RW	1	Memory wait states for the slow clock domain ('clk_slow'). The number of wait states is expressed in 'clk_hf' clock domain cycles. Timing paths to and from the memory have a (fixed) minimum duration that always needs to be considered/met. The 'clk_hf' clock domain frequency determines this field's value such that the timing paths minimum duration is met. ROM_CTL.SLOW_WS = '0' when clk_hf <=100 MHz. ROM_CTL.SLOW_WS = '1' when 100MHz < clk_hf <=clk_hf_max. Note: clk_hf_max depends on the target device. Refer datasheet.
8:9	FAST_WS	RW	RW	0	Memory wait states for the fast clock domain ('clk_fast'). The number of wait states is expressed in 'clk_hf' clock domain cycles. ROM_CTL.FAST_WS = '0' when clk_hf <= clk_hf_max.

```
/************ Setting wait state for RAM **********/
CPUSS->unRAM0_CTL0.stcField.u2SLOW_WS = 1u;
CPUSS->unRAM0_CTL0.stcField.u2FAST_WS = 0u;
```

Bits	Name	SW	HW	Default or Enum	Description
0:1	SLOW_WS	RW	RW	1	Memory wait states for the slow clock domain ('clk_slow'). The number of wait states is expressed in 'clk_hf' clock domain cycles.
8:9	FAST_WS	RW	RW	0	Memory wait states for the fast clock domain ('clk_fast'). The number of wait states is expressed in 'clk_hf' clock domain cycles.
16	ECC_EN	RW	RW	1	Enable ECC checking: '0': Disabled. '1': Enabled.
17	ECC_AUTO_CORRECT	RW	RW	1	HW ECC autocorrect functionality: '0': Disabled. '1': Enabled. HW automatically writes back SRAM with corrected data when a recoverable ECC error is detected.
18	ECC_INJ_EN	RW	RW	0	Enable error injection for system SRAM 0. When '1', the parity (ECC_CTL.PARITY) is used when a full 32-bit write is done to the ECC_CTL.WORD_ADDR word address of system SRAM 0.

```
  #if defined (CPUSS_RAMC1_PRESENT) && (CPUSS_RAMC1_PRESENT == 1UL)
    CPUSS->unRAM1_CTL0.stcField.u2SLOW_WS = 1u;
    CPUSS->unRAM1_CTL0.stcField.u2FAST_WS = 0u;
  #endif /* defined (CPUSS_RAMC1_PRESENT) && (CPUSS_RAMC1_PRESENT
== 1UL) */
```

Bits	Name	SW	HW	Default or Enum	Description
0:1	SLOW_WS	RW	RW	1	Memory wait states for the slow clock domain ('clk_slow'). The number of wait states is expressed in 'clk_hf' clock domain cycles.
8:9	FAST_WS	RW	RW	0	Memory wait states for the fast clock domain ('clk_fast'). The number of wait states is expressed in 'clk_hf' clock domain cycles.
16	ECC_EN	RW	RW	1	Enable ECC checking: '0': Disabled. '1': Enabled.
17	ECC_AUTO_CORRECT	RW	RW	1	HW ECC autocorrect functionality: '0': Disabled. '1': Enabled. HW automatically writes back SRAM with corrected data when a recoverable ECC error is detected.
18	ECC_INJ_EN	RW	RW	0	Enable error injection for system SRAM 0. When '1', the parity (ECC_CTL.PARITY) is used when a full 32-bit write is done to the ECC_CTL.WORD_ADDR word address of system SRAM 0.

```
  #if defined (CPUSS_RAMC2_PRESENT) && (CPUSS_RAMC2_PRESENT == 1UL)
    CPUSS->unRAM2_CTL0.stcField.u2SLOW_WS = 1u;
    CPUSS->unRAM2_CTL0.stcField.u2FAST_WS = 0u;
  #endif /* defined (CPUSS_RAMC2_PRESENT) && (CPUSS_RAMC2_PRESENT
== 1UL) */
```

Not Present in TVII 1M STARTER KIT

```
    /************ Setting wait state for FLASH **********/
    FLASHC->unFLASH_CTL.stcField.u4MAIN_WS = 1u;
```

Bits	Name	SW	HW	Default or Enum	Description
0:3	MAIN_WS	RW	RW	0	FLASH macro main interface wait states: '0': 0 wait states. ... '15': 15 wait states
8	MAIN_MAP	RW	RW	0	Specifies mapping of FLASH macro main array. 0: Mapping A. 1: Mapping B. This field is only used when MAIN_BANK_MODE is '1' (dual bank mode).

```
    /***    Set clock LF source         ***/
 SRSS->unCLK_SELECT.stcField.u3LFCLK_SEL = CY_SYSCLK_LFCLK_IN_ILO0;
    /* CY_SYSCLK_LFCLK_IN_ILO0 = 0 */
```

Bits	Name	SW	HW	Default or Enum	Description
0:2	LFCLK_SEL	RW	RW	0	Select source for LFCLK. Note that not all products support all clock sources. Selecting a clock source that is not supported will result in undefined behavior. Writes to this field are ignored unless the WDT is unlocked using WDT_LOCK register. It takes four cycles of the originally selected clock to switch away from it. Do not disable the original clock during this time.

```
#if CY_SYSTEM_USE_CLOCK == CY_SYSTEM_USE_ECO

    /***    ECO port settings         ***/
    /* Default settings should be OK. */

    /***    ECO setting and enabling         ***/
    // These values need to be confirmed
    SRSS->unCLK_ECO_CONFIG2.stcField.u3WDTRIM = 4u;
    SRSS->unCLK_ECO_CONFIG2.stcField.u4ATRIM  = 12u;
    SRSS->unCLK_ECO_CONFIG2.stcField.u2FTRIM  = 3u;
    SRSS->unCLK_ECO_CONFIG2.stcField.u2RTRIM  = 3u;
    SRSS->unCLK_ECO_CONFIG2.stcField.u3GTRIM  = 1u;
    SRSS->unCLK_ECO_CONFIG.stcField.u1ECO_EN = 1ul;
    while(SRSS->unCLK_ECO_STATUS.stcField.u1ECO_OK == 0ul);
    while(SRSS->unCLK_ECO_STATUS.stcField.u1ECO_READY == 0ul);
#endif
```

It is not applicable for Starter kit (Starter kit uses IMO Frequency)

```
#if defined(CY_SYSTEM_WCO_ENABLE)

    // Enable WCO
    BACKUP->unCTL.stcField.u1WCO_EN = 1ul;
    // Wait until WCO status becomes OK
    while(BACKUP->unSTATUS.stcField.u1WCO_OK == 0ul);

#endif
```

It is also not true, so it also not be executed

```
    /***  Set CPUSS dividrs as required        ***/
    // FAST = 160,000,000; PERI and SLOW = FAST / 2;
    CPUSS->unCM4_CLOCK_CTL.stcField.u8FAST_INT_DIV = 0u; // no
division
```

Bits	Name	SW	HW	Default or Enum	Description
8:15	FAST_INT_DIV	RW	RW	0	Specifies the fast clock divider (from the high frequency clock 'clk_hf' to the peripheral clock 'clk_fast'). Integer division by (1+FAST_INT_DIV). Allows for integer divisions in the range [1, 256] (FAST_INT_DIV is in the range [0, 255]). Note that this field is retained. However, the counter that is used to implement the division is not and will be initialized by HW to '0' when transitioning from DeepSleep to Active power mode.

```
CPUSS->unCM0_CLOCK_CTL.stcField.u8PERI_INT_DIV = 1u; //divided by 2
CPUSS->unCM0_CLOCK_CTL.stcField.u8SLOW_INT_DIV = 0u; //no division
```

Bits	Name	SW	HW	Default or Enum	Description
8:15	SLOW_INT_DIV	RW	RW	0	Specifies the slow clock divider (from the peripheral clock 'clk_peri' to the slow clock 'clk_slow'). Integer division by (1+SLOW_INT_DIV). Allows for integer divisions in the range [1, 256] (SLOW_INT_DIV is in the range [0, 255]). Note that this field is retained. However, the counter that is used to implement the division is not and will be initialized by HW to '0' when transitioning from DeepSleep to Active power mode.
24:31	PERI_INT_DIV	RW	RW	0	Specifies the peripheral clock divider (from the high frequency clock 'clk_hf' to the peripheral clock 'clk_peri'). Integer division by (1+PERI_INT_DIV). Allows for integer divisions in the range [1, 256] (PERI_INT_DIV is in the range [0, 255]). Note that this field is retained. However, the counter that is used to implement the division is not and will be initialized by HW to '0' when transitioning from DeepSleep to Active power mode. Note that Fperi <= Fperi_max. Fperi_max is likely to be smaller than Fhf_max. In other words, if Fhf = Fhf_max, PERI_INT_DIV should not be set to '0'.

```
    /***     PLL setting and enabling        ***/
    SRSS->unCLK_PATH_SELECT[1/*PLL0*/].stcField.u3PATH_MUX =
          CY_SYSTEM_PLL_INPUT_SOURCE;
    /* CY_SYSTEM_PLL_INPUT_SOURCE = 0 */
```

Bits	Name	SW	HW	Default or Enum	Description
0:2	PATH_MUX	RW	RW	0	Selects a source for clock PATH<i>. Note that not all products support all clock sources. Selecting a clock source that is not supported will result in undefined behavior. It takes four cycles of the originally selected clock to switch away from it. Do not disable the original clock during this time.
	IMO			0	IMO - Internal R/C Oscillator
	EXTCLK			1	EXTCLK - External Clock Pin
	ECO			2	ECO - External-Crystal Oscillator
	ALTHF			3	ALTHF - Alternate High-Frequency clock input (product-specific clock)
	DSI_MUX			4	DSI_MUX - Output of DSI mux for this path. Using a DSI source directly as root of HFCLK will result in undefined behavior.
	LPECO			5	N/A

```
SRSS->unCLK_PLL_CONFIG[0].stcField.u5REFERENCE_DIV      =
      CY_SYSTEM_PLL_CONFIG_REFDIV;
 /* CY_SYSTEM_PLL_CONFIG_REFDIV = 1 */
```

8:12	REFERENCE_DIV	RW	RW	1	Control bits for reference divider. Set the divide value before enabling the PLL, and do not change it while PLL is enabled. 0: illegal (undefined behavior) 1: divide by 1 ... 20: divide by 20 others: illegal (undefined behavior)

```
SRSS->unCLK_PLL_CONFIG[0].stcField.u7FEEDBACK_DIV       =
      CY_SYSTEM_PLL_CONFIG_FEEDBACKDIV;
/* CY_SYSTEM_PLL_CONFIG_FEEDBACKDIV = 40 */
```

0:6	FEEDBACK_DIV	RW	RW	22	Control bits for feedback divider. Set the divide value before enabling the PLL, and do not change it while PLL is enabled. 0-21: illegal (undefined behavior) 22: divide by 22 ... 112: divide by 112 >112: illegal (undefined behavior)

```
SRSS->unCLK_PLL_CONFIG[0].stcField.u5OUTPUT_DIV         =
      CY_SYSTEM_PLL_CONFIG_OUTDIV;
/* CY_SYSTEM_PLL_CONFIG_OUTDIV = 2 */
```

16:20	OUTPUT_DIV	RW	RW	2	Control bits for Output divider. Set the divide value before enabling the PLL, and do not change it while PLL is enabled. 0: illegal (undefined behavior) 1: illegal (undefined behavior) 2: divide by 2. Suitable for direct usage as HFCLK source. ... 16: divide by 16. Suitable for direct usage as HFCLK source. >16: illegal (undefined behavior)

```
SRSS->unCLK_PLL_CONFIG[0].stcField.u1ENABLE                = 1ul;
```

31	ENABLE	RW	RW	0	Master enable for PLL. Setup FEEDBACK_DIV, REFERENCE_DIV, and OUTPUT_DIV at least one cycle before setting ENABLE=1. Fpll = (FEEDBACK_DIV) * (Fref / REFERENCE_DIV) / (OUTPUT_DIV) 0: Block is disabled. When the PLL disables, hardware controls the bypass mux as described in BYPASS_SEL, before disabling the PLL circuit. 1: Block is enabled

```
while(SRSS->unCLK_PLL_STATUS[0].stcField.u1LOCKED == 0ul);
```

Bits	Name	SW	HW	Default or Enum	Description
0	LOCKED	R	R	0	PLL Lock Indicator
1	UNLOCK_OCCURRED	RW1C	RW1C	0	This bit sets whenever the PLL Lock bit goes low, and stays set until cleared by firmware.

```
    /***   Setting  PATH2  source          ***/
    SRSS->unCLK_PATH_SELECT[2].stcField.u3PATH_MUX =
          CY_SYSCLK_CLKPATH_IN_IMO;
    /* CY_SYSCLK_CLKPATH_IN_IMO = 0 */
```

Bits	Name	SW	HW	Default or Enum	Description
0:2	PATH_MUX	RW	RW	0	Selects a source for clock PATH<i>. Note that not all products support all clock sources. Selecting a clock source that is not supported will result in undefined behavior. It takes four cycles of the originally selected clock to switch away from it. Do not disable the original clock during this time.
	IMO			0	IMO - Internal R/C Oscillator
	EXTCLK			1	EXTCLK - External Clock Pin
	ECO			2	ECO - External-Crystal Oscillator
	ALTHF			3	ALTHF - Alternate High-Frequency clock input (product-specific clock)
	DSI_MUX			4	DSI_MUX - Output of DSI mux for this path. Using a DSI source directly as root of HFCLK will result in undefined behavior.
	LPECO			5	N/A

```
/***  Assign  PLL0 as source of clk_hf0          ***/
/* Select source of clk_hf0 *//* Set HF source, divider, enable */
   SRSS->unCLK_ROOT_SELECT[0/*clk_hf0*/].stcField.u4ROOT_MUX =
         CY_SYSCLK_HFCLK_IN_CLKPATH1;
   /* CY_SYSCLK_HFCLK_IN_CLKPATH1 = 1 */
```

Bits	Name	SW	HW	Default or Enum	Description
0:3	ROOT_MUX	RW	RW	0	Selects a clock path as the root of HFCLK<k> and for SRSS DSI input <k>. Use CLK_PATH_SELECT[i] to configure the desired path. Some paths may have FLL or PLL available (product-specific), and the control and bypass mux selections of these are in other registers. Configure the FLL using CLK_FLL_CONFIG register. Configure a PLL using the related CLK_PLL_CONFIG[k] register. Note that not all products support all clock sources. Selecting a clock source that is not supported will result in undefined behavior. It takes four cycles of the originally selected clock to switch away from it. Do not disable the original clock during this time.
	PATH0			0	Select PATH0 (can be configured for FLL)
	PATH1			1	Select PATH1 (can be configured for PLL0, if available in the product)
	PATH2			2	Select PATH2 (can be configured for PLL1, if available in the product)
	PATH3			3	Select PATH3 (can be configured for PLL2, if available in the product)
	PATH4			4	Select PATH4 (can be configured for PLL3, if available in the product)
	PATH5			5	Select PATH5 (can be configured for PLL4, if available in the product)
	PATH6			6	Select PATH6 (can be configured for PLL5, if available in the product)
	PATH7			7	Select PATH7 (can be configured for PLL6, if available in the product)
	PATH8			8	Select PATH8 (can be configured for PLL7, if available in the product)
	PATH9			9	Select PATH9 (can be configured for PLL8, if available in the product)
	PATH10			10	Select PATH10 (can be configured for PLL9, if available in the product)
	PATH11			11	Select PATH11 (can be configured for PLL10, if available in the product)
	PATH12			12	Select PATH12 (can be configured for PLL11, if available in the product)
	PATH13			13	Select PATH13 (can be configured for PLL12, if available in the product)
	PATH14			14	Select PATH14 (can be configured for PLL13, if available in the product)
	PATH15			15	Select PATH15 (can be configured for PLL14, if available in the product)

```
SRSS->unCLK_ROOT_SELECT[0].stcField.u2ROOT_DIV = 0u; /* no div */
SRSS->unCLK_ROOT_SELECT[0].stcField.u1ENABLE   = 1u; /* 1=enable */
```

4:5	ROOT_DIV	RW	RW	0	Selects predivider value for this clock root and DSI input.
	NO_DIV			0	Transparent mode, feed through selected clock source w/o dividing.
	DIV_BY_2			1	Divide selected clock source by 2
	DIV_BY_4			2	Divide selected clock source by 4
	DIV_BY_8			3	Divide selected clock source by 8
31	ENABLE	RW	RW	0	Enable for this clock root. All clock roots default to disabled (ENABLE==0) except HFCLK0, which cannot be disabled.

```
    /* Select source of clk_hf1 */
    /***  Set HF1 source, divider, enable   ***/
    SRSS->unCLK_ROOT_SELECT[1/*clk_hf1*/].stcField.u4ROOT_MUX =
CY_SYSCLK_HFCLK_IN_CLKPATH1;
    /* CY_SYSCLK_HFCLK_IN_CLKPATH1 = 1 */
```

Bits	Name	SW	HW	Default or Enum	Description
0:3	ROOT_MUX	RW	RW	0	Selects a clock path as the root of HFCLK<k> and for SRSS DSI input <k>. Use CLK_PATH_SELECT[i] to configure the desired path. Some paths may have FLL or PLL available (product-specific), and the control and bypass mux selections of these are in other registers. Configure the FLL using CLK_FLL_CONFIG register. Configure a PLL using the related CLK_PLL_CONFIG[k] register. Note that not all products support all clock sources. Selecting a clock source that is not supported will result in undefined behavior. It takes four cycles of the originally selected clock to switch away from it. Do not disable the original clock during this time.
	PATH0			0	Select PATH0 (can be configured for FLL)
	PATH1			1	Select PATH1 (can be configured for PLL0, if available in the product)
	PATH2			2	Select PATH2 (can be configured for PLL1, if available in the product)
	PATH3			3	Select PATH3 (can be configured for PLL2, if available in the product)
	PATH4			4	Select PATH4 (can be configured for PLL3, if available in the product)
	PATH5			5	Select PATH5 (can be configured for PLL4, if available in the product)
	PATH6			6	Select PATH6 (can be configured for PLL5, if available in the product)
	PATH7			7	Select PATH7 (can be configured for PLL6, if available in the product)
	PATH8			8	Select PATH8 (can be configured for PLL7, if available in the product)
	PATH9			9	Select PATH9 (can be configured for PLL8, if available in the product)
	PATH10			10	Select PATH10 (can be configured for PLL9, if available in the product)
	PATH11			11	Select PATH11 (can be configured for PLL10, if available in the product)
	PATH12			12	Select PATH12 (can be configured for PLL11, if available in the product)
	PATH13			13	Select PATH13 (can be configured for PLL12, if available in the product)
	PATH14			14	Select PATH14 (can be configured for PLL13, if available in the product)
	PATH15			15	Select PATH15 (can be configured for PLL14, if available in the product)

```
    SRSS->unCLK_ROOT_SELECT[1].stcField.u2ROOT_DIV = 1u; /* divided
by 2 */
    SRSS->unCLK_ROOT_SELECT[1].stcField.u1ENABLE   = 1u; /* 1 =
enable */
```

4:5	ROOT_DIV	RW	RW	0	Selects predivider value for this clock root and DSI input.
	NO_DIV			0	Transparent mode, feed through selected clock source w/o dividing.
	DIV_BY_2			1	Divide selected clock source by 2
	DIV_BY_4			2	Divide selected clock source by 4
	DIV_BY_8			3	Divide selected clock source by 8
31	ENABLE	RW	RW	0	Enable for this clock root. All clock roots default to disabled (ENABLE==0) except HFCLK0, which cannot be disabled.

```
    /***     Enabling ILO0         ***/
    Cy_WDT_Unlock();  // Unlock the WatchDog Timer
    SRSS->unCLK_ILO0_CONFIG.stcField.u1ENABLE = 1; /* 1 = enable */
    SRSS->unCLK_ILO0_CONFIG.stcField.u1ILO0_BACKUP = 1ul; /* Ilo
HibernateOn */
    Cy_WDT_Lock();   // Lock the WatchDog Timer
```

Bits	Name	SW	HW	Default or Enum	Description
0	ILO0_BACKUP	RW	RW	0	This register indicates that ILO0 should stay enabled during XRES and HIBERNATE modes. If backup voltage domain is implemented on the product, this bit also indicates if ILO0 should stay enabled through power-related resets on other supplies, e.g.. BOD on VDDD/VCCD. Writes to this field are ignored unless the WDT is unlocked using WDT_LOCK register. This register is reset when the backup logic resets. 0: ILO0 turns off during XRES, HIBERNATE, and power-related resets. ILO0 configuration and trims are reset by these events. 1: ILO0 stays enabled, as described above. ILO0 configuration and trims are not reset by these events.
30	ILO0_MON_ENABLE	RW	RW	0	Reserved. Always write a zero. Writes to this field are ignored unless the WDT is unlocked using WDT_LOCK register.
31	ENABLE	RW	RW	1	Master enable for ILO. Writes to this field are ignored unless the WDT is unlocked using WDT_LOCK register. HT-variant: This register will not clear unless PWR_CTL2.BGREF_LPMODE==0. After enabling, the first ILO0 cycle occurs within 12us and is +/-10 percent accuracy. Thereafter, ILO0 is +/-5 percent accurate.

For 1M starter kit (PLL0)

Reference Div = 1
Feedback Div = 40
Output Div = 2

$PLL_{out} = (Feedback\ Div) * (PLL_{in} / Reference\ Div) / Output\ Div$

$= 40 * (8 \times 10^6) / 2$

$= 160 \times 10^6$

$= 160\ MHz$

7. CYT2B7 Clock Diagram

Figure 7-1.CYT2B7 Clock Diagram