**<trimLPO()>**

```
if(LPO_TRIM_VALUE != 0xFFFFU) //ffff가 아니면 구동이 된다. LPO는 저전력 발진기.
    {

        systemREG1->LPOMONCTL   = (uint32)((uint32)1U << 24U) //24번비트를 사용하겠다.
```

**HF는 중심 주파수라고 해서 우리가 원하는 주파수로 깍아 버린다는 것이다.**

```
                                    | LPO_TRIM_VALUE;
    }
    else
    {
        systemREG1->LPOMONCTL    =   (uint32)((uint32)1U << 24U)
                                    | (uint32)((uint32)16U << 8U)
                                    | 16U;
    }
```

| Bit | Field | Value | Description |
|---|---|---|---|
| 12-8 | HFTRIM | | High-frequency oscillator trim value. This four-bit value is used to center the HF oscillator's frequency.<br>**Caution: This value should only be changed when the HF oscillator is not the source for a clock domain, otherwise a system failure could result.**<br>The following values are the ratio: f / fo in the F021 process. |

### 2.5.1.30 LPO/Clock Monitor Control Register (LPOMONCTL)

The LPOMONCTL register, shown in Figure 2-37 and described in Table 2-49, controls the Low Frequency (Clock Source 4) and High Frequency (Clock Source 5) Low Power Oscillator's trim values.

**Figure 2-37. LPO/Clock Monitor Control Register (LPOMONCTL) (offset = 088h)**

| 31 | 25 | 24 | 23 | 17 | 16 |
|---|---|---|---|---|---|
| Reserved | | BIAS ENABLE | Reserved | | OSCFRQCONFIGCNT |
| R-0 | | R/WP-1 | R-0 | | R/WP-0 |

| 15 | 13 | 12 | 8 | 7 | 5 | 4 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | HFTRIM | | Reserved | | LFTRIM | |
| R-0 | | R/WP-10h | | R-0 | | R/WP-10h | |

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

**Table 2-49. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-25 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 24 | BIAS ENABLE | | Bias enable. |
| | | 0 | The bias circuit inside the low-power oscillator (LPO) is disabled. |
| | | 1 | The bias circuit inside the low-power oscillator (LPO) is enabled. |
| 23-17 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 16 | OSCFRQCONFIGCNT | | Configures the counter based on OSC frequency. |
| | | 0 | *Read:* OSC freq is ≤ 20MHz. |
| | | | *Write:* A write of 0 has no effect. |
| | | 1 | *Read:* OSC freq is > 20MHz and ≤ 80MHz. |
| | | | *Write:* A write of 1 has no effect. |
| 15-13 | Reserved | 0 | Reads return 0. Writes have no effect. |

**#define LPO_TRIM_VALUE (((*(volatile uint32 *)0xF00801B4U) & 0xFFFF0000U)>>16U)**

**:최상위 16비트만 사용하겠다.**

**/* USER CODE BEGIN (11) */**

**/* USER CODE END */**

```
/** @b Initialize @b Clock @b Tree: */

/** - Setup system clock divider for HCLK */

systemREG2->HCLKCNTL = 1U;//2분주를 시켜서 사용 ,상당히 통신하는데 빠르다.(300MHZ)


/** - Disable / Enable clock domain */

systemREG1->CDDIS = (uint32)((uint32)0U << 4U ) /* AVCLK1 , 1 - OFF, 0 - ON */ //4번 비트를 사용

                  | (uint32)((uint32)1U << 5U ) /* AVCLK2 , 1 - OFF, 0 - ON */
```

**| (uint32)((uint32)0U << 8U ) /* VCLK3 , 1 - OFF, 0 - ON */**

**| (uint32)((uint32)0U << 9U ) /* VCLK4 , 1 - OFF, 0 - ON */**

**| (uint32)((uint32)0U << 10U) /* AVCLK3 , 1 - OFF, 0 - ON */**

**| (uint32)((uint32)0U << 11U); /* AVCLK4 , 1 - OFF, 0 - ON */**

---

**Figure 2-20. Clock Domain Disable Register (CDDIS) (offset = 3Ch)**

| 31 | | | | | | | 16 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| R-0 | | | | | | | |
| 15 | | | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | VCLKA4OFF | Reserved | Reserved | VCLK3OFF |
| R-0 | | | | R/WP-0 | R/WP-0 | R/WP-0 | R/WP-0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RTICLK1OFF | VCLKA2OFF | VCLKA1OFF | VCLK2OFF | VCLKPOFF | HCLKOFF | GCLK1OFF |
| R/WP-0 | R/WP-0 | R/WP-0 | R/WP-0 | R/WP-0 | R/WP-0 | R/WP-0 | R/WP-0 |

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

**Table 2-32. Clock Domain Disable Register (CDDIS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-12 | Reserved | 0-1 | Reads return 0 or 1 and privilege mode writes allowed. |
| 11 | VCLKA4OFF | | VCLKA4 domain off. |
| | | 0 | The VCLKA4 domain is enabled. |
| | | 1 | The VCLKA4 domain is disabled. |
| 10-9 | Reserved | 0-1 | Reads return 0 or 1 and privilege mode writes allowed. |
| 8 | VCLK3OFF | | VCLK3 domain off. |
| | | 0 | The VCLK3 domain is enabled. |
| | | 1 | The VCLK3 domain is disabled. |
| 7 | Reserved | 0-1 | Reads return 0 or 1 and privilege mode writes allowed. |
| 6 | RTICLK1OFF | | RTICLK1 domain off. |
| | | 0 | The RTICLK1 domain is enabled. |
| | | 1 | The RTICLK1 domain is disabled. |
| 5-4 | VCLKA[2-1]OFF | | VCLKA[2-1] domain off. |
| | | 0 | The VCLKA[2-1] domain is enabled. |
| | | 1 | The VCLKA[2-1] domain is disabled. |
| 3 | VCLK2OFF | | VCLK2 domain off. |
| | | 0 | The VCLK2 domain is enabled. |
| | | 1 | The VCLK2 domain is disabled. |
| 2 | VCLKPOFF | | VCLK_periph domain off. |
| | | 0 | The VCLK_periph domain is enabled. |
| | | 1 | The VCLK_periph domain is disabled. |

## 5.5 Switching Characteristics over Recommended Operating Conditions for Clock Domains

**Table 5-2. Clock Domain Timing Specifications**

| | PARAMETER | TEST CONDITIONS | MIN | MAX | UNIT |
|---|---|---|---|---|---|
| $f_{OSC}$ | OSC - oscillator clock frequency using an external crystal | | 5 | 20 | MHz |
| $f_{GCLK1}$ | GCLK - R5F CPU clock frequency | | | 300 | MHz |
| $f_{GCLK2}$ | GCLK - R5F CPU clock frequency | | | 300 | MHz |
| $f_{HCLK}$ | HCLK - System clock frequency | | | 150 | MHz |
| $f_{VCLK}$ | VCLK - Primary peripheral clock frequency | | | 110 | MHz |
| $f_{VCLK2}$ | VCLK2 - Secondary peripheral clock frequency | | | 110 | MHz |
| $f_{VCLK3}$ | VCLK3 - Secondary peripheral clock frequency | | | 150 | MHz |
| $f_{VCLKA1}$ | VCLKA1 - Primary asynchronous peripheral clock frequency | | | 110 | MHz |
| $f_{VCLKA2}$ | VCLKA2 - Secondary asynchronous peripheral clock frequency | | | 110 | MHz |
| $f_{VCLKA4}$ | VCLKA4 - Secondary asynchronous peripheral clock frequency | | | 110 | MHz |
| $f_{RTICLK1}$ | RTICLK1 - clock frequency | | | $f_{VCLK}$ | MHz |
| $f_{PROG/ERASE}$ | System clock frequency - flash programming/erase | | | $f_{HCLK}$ | MHz |
| $f_{ECLK1}$ | External Clock 1 | | | 110 | MHz |
| $f_{ECLK2}$ | External Clock 2 | | | 110 | MHz |
| $f_{ETMCLKOUT}$ | ETM trace clock output | | | 55 | MHz |
| $f_{ETMCLKIN}$ | ETM trace clock input | | | 110 | MHz |
| $f_{EXTCLKIN1}$ | External input clock 1 | | | 110 | MHz |
| $f_{EXTCLKIN2}$ | External input clock 2 | | | 110 | MHz |

각 VCLKA마다 사용하는 주파수가 다르다.

### 2.5.1.10 Clock Source Disable Register (CSDIS)

The CSDIS register, shown in Figure 2-17 and described in Table 2-28, controls and displays the state of the device clock sources.

**Figure 2-17. Clock Source Disable Register (CSDIS) (offset = 30h)**

| 31 | | | | | | | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| R-0 | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLKSR7OFF | CLKSR6OFF | CLKSR5OFF | CLKSR4OFF | CLKSR3OFF | Reserved | CLKSR1OFF | CLKSR0OFF |
| R/WP-1 | R/WP-1 | R/WP-0 | R/WP-0 | R/WP-1 | R-1 | R/WP-1 | R/WP-0 |

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; *-n* = value after reset

**Table 2-28. Clock Source Disable Register (CSDIS) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 7-3 | CLKSR[7-3]OFF | | Clock source[7-3] off. |
| | | 0 | Clock source[7-3] is enabled. |
| | | 1 | Clock source[7-3] is disabled. |
| | | | **Note: On wakeup, only clock sources 0, 4, and 5 are enabled.** |
| 2 | Reserved | 1 | Reads return 1. Writes have no effect. |
| 1-0 | CLKSR[1-0]OFF | | Clock source[1-0] off. |
| | | 0 | Clock source[1-0] is enabled. |
| | | 1 | Clock source[1-0] is disabled. |
| | | | **Note: On wakeup, only clock sources 0, 4, and 5 are enabled.** |

**Table 2-29. Clock Sources Table**

| Clock Source # | Clock Source Name |
|---|---|
| Clock Source 0 | Oscillator |
| Clock Source1 | PLL1 |
| Clock Source 2 | Not Implemented |
| Clock Source 3 | EXTCLKIN |
| Clock Source 4 | Low Frequency LPO (Low Power Oscillator) clock |
| Clock Source 5 | High frequency LPO (Low Power Oscillator) clock |
| Clock Source 6 | PLL2 |
| Clock Source 7 | EXTCLKIN2 |

### 2.5.1.13 Clock Domain Disable Register (CDDIS)

The CDDIS register, shown in Figure 2-20 and described in Table 2-32, controls the state of the clock domains.

**NOTE: All the clock domains are enabled on wakeup.**

The application should assure that when HCLK and VCLK_sys are turned off through the HCLKOFF bit, the GCLK1 domain is also turned off.

The register bits in CDDIS are designated as high-integrity bits and have been implemented with error-correcting logic such that each bit, although read and written as a single bit, is actually a multi-bit key with error correction capability. As such, single-bit flips within the "key" can be corrected allowing protection of the system as a whole. An error detected is signaled to the ESM module.

**HTU는 초정밀 타이머를 동작할 수 있다.(n(나노)t까지)**

<SYSTEM 안정화>

```
/** - Wait for until clocks are locked */
    SYS_CSVSTAT = systemREG1->CSVSTAT;
    SYS_CSDIS = systemREG1->CSDIS;
```

여기서 LOCK이란 무엇인가?

처음에 OVERFLOW가 생기면 오버플로우가 안정이 될때까지 딜레이 시간이 필요하기에 그전까지 LOCK을 걸어놓는다.

```
systemREG1->GHVSRC = (uint32)((uint32)SYS_PLL1 << 24U)
                   | (uint32)((uint32)SYS_PLL1 << 16U)
                   | (uint32)((uint32)SYS_PLL1 << 0U);
```

<real-time컴파일러 최적화를 위한 코딩 = 쓸때없는 코드>

```
  systemREG1->RCLKSRC = (uint32)((uint32)1U << 24U)          /* RTI2 divider (Not applicable for lock-step device) */
                      | (uint32)((uint32)SYS_VCLK << 16U) /* RTI2 clock source (Not applicable for lock-step device) */
                      | (uint32)((uint32)1U << 8U)          /* RTI1 divider */
                      | (uint32)((uint32)SYS_VCLK << 0U);

  systemREG1->VCLKASRC = (uint32)((uint32)SYS_VCLK << 8U)
                       | (uint32)((uint32)SYS_VCLK << 0U);
```

:동기화

```
systemREG1->CLKCNTL  = (systemREG1->CLKCNTL & 0xF0FFFFFFU)
                     | (uint32)((uint32)1U << 24U);
```

VCLK2R의 2분주가 되겠다.

```
systemREG2->CLK2CNTRL = (systemREG2->CLK2CNTRL & 0xFFFFFFF0U)
                      | (uint32)((uint32)1U << 0U);
```

:vclk3를 2분주로 사용하겠다는 의미

```
systemREG2->VCLKACON1 =  (uint32)((uint32)(1U - 1U) << 24U)
                      | (uint32)((uint32)0U << 20U)
                      | (uint32)((uint32)SYS_VCLK << 16U) // VCLKA4와VCLK의 속도가 같다.
                      | (uint32)((uint32)(1U - 1U) << 8U)
                      | (uint32)((uint32)0U << 4U)
                      | (uint32)((uint32)SYS_VCLK << 0U);
```

### 2.5.2.4 Clock 2 Control Register (CLK2CNTRL)

This register is shown in Figure 2-62 and described in Table 2-75.

**Figure 2-62. Clock 2 Control Register (CLK2CNTRL) (offset = 3Ch)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 12 | 11 | 8 | 7 | 4 | 3 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | Reserved | | Reserved | | VCLK3R | |
| R-0 | | R/WP-1h | | R-0 | | R/WP-1h | |

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

**Table 2-75. Clock 2 Control Register (CLK2CNTRL) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-12 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 11-8 | Reserved | | Reads return value and writes allowed in privilege mode. |
| 7-4 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 3-0 | VCLK3R | | VBUS clock3 ratio. |
| | | 0 | The ratio is HCLK divide by 1. |
| | | : | : |
| | | Fh | The ratio is HCLK divided by 16. |

### 2.5.1.16 GCLK1, HCLK, VCLK, and VCLK2 Source Register (GHVSRC)

The GHVSRC register, shown in Figure 2-23 and described in Table 2-35, controls the clock source configuration for the GCLK1, HCLK, VCLK and VCLK2 clock domains.

**Figure 2-23. GCLK1, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) (offset = 48h)**

| 31 28 | 27 24 | 23 20 | 19 16 |
|---|---|---|---|
| Reserved | GHVWAKE | Reserved | HVLPM |
| R-0 | R/WP-0 | R-0 | R/WP-0 |

| 15 4 | 3 0 |
|---|---|
| Reserved | GHVSRC |
| R-0 | R/WP-0 |

LEGEND: R = Read only; R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

**Table 2-35. GCLK1, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-28 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 27-24 | GHVWAKE | | GCLK1, HCLK, VCLK source on wakeup. |
| | | 0 | Clock source0 is the source for GCLK1, HCLK, VCLK on wakeup. |
| | | 1h | Clock source1 is the source for GCLK1, HCLK, VCLK on wakeup. |
| | | 2h | Clock source2 is the source for GCLK1, HCLK, VCLK on wakeup. |
| | | 3h | Clock source3 is the source for GCLK1, HCLK, VCLK on wakeup. |
| | | 4h | Clock source4 is the source for GCLK1, HCLK, VCLK on wakeup. |
| | | 5h | Clock source5 is the source for GCLK1, HCLK, VCLK on wakeup. |
| | | 6h | Clock source6 is the source for GCLK1, HCLK, VCLK on wakeup. |
| | | 7h | Clock source7 is the source for GCLK1, HCLK, VCLK on wakeup. |
| | | 8h-Fh | Reserved |
| 23-20 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 19-16 | HVLPM | | HCLK, VCLK, VCLK2 source on wakeup when GCLK1 is turned off. |
| | | 0 | Clock source0 is the source for HCLK, VCLK, VCLK2 on wakeup. |
| | | 1h | Clock source1 is the source for HCLK, VCLK, VCLK2 on wakeup. |
| | | 2h | Clock source2 is the source for HCLK, VCLK, VCLK2 on wakeup. |
| | | 3h | Clock source3 is the source for HCLK, VCLK, VCLK2 on wakeup. |
| | | 4h | Clock source4 is the source for HCLK, VCLK, VCLK2 on wakeup. |
| | | 5h | Clock source5 is the source for HCLK, VCLK, VCLK2 on wakeup. |
| | | 6h | Clock source6 is the source for HCLK, VCLK, VCLK2 on wakeup. |
| | | 7h | Clock source7 is the source for HCLK, VCLK, VCLK2 on wakeup. |
| | | 8h-Fh | Reserved |

```
/* Now the PLLs are locked and the PLL outputs can be sped up */

    /* The R-divider was programmed to be 0xF. Now this divider is changed to programmed value */

    systemREG1->PLLCTL1 = (systemREG1->PLLCTL1 & 0xE0FFFFFFU) | (uint32)((uint32)(1U - 1U) << 24U); //24번비트를 분주없이 설정해놓았다는것을 의미

    /*SAFETYMCUSW 134 S MR:12.2 <APPROVED> " Clear and write to the volatile register " */

    systemREG2->PLLCTL3 = (systemREG2->PLLCTL3 & 0xE0FFFFFFU) | (uint32)((uint32)(1U -
```

1U) << 24U); //이하동문

```
systemREG1->SYSPC1 = 0U;    // ECLK가 GIO모드로 동작을 한다.
    /** - set ECLK pins default output value */
    systemREG1->SYSPC4 = 0U;
    /** - set ECLK pins output direction */
    systemREG1->SYSPC2 = 1U; //출력모드
    /** - set ECLK pins open drain enable */
    systemREG1->SYSPC7 = 0U;
    /** - set ECLK pins pullup/pulldown enable */
    systemREG1->SYSPC8 = 0U;
/** - set ECLK pins pullup/pulldown select */
    systemREG1->SYSPC9 = 1U;
```

#### 2.5.1.1 SYS Pin Control Register 1 (SYSPC1)

The SYSPC1 register, shown in Figure 2-8 and described in Table 2-19, controls the function of the ECLK pin.

**Figure 2-8. SYS Pin Control Register 1 (SYSPC1) (offset = 00h)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 1 | 0 |
|---|---|---|
| Reserved | | ECPCLKFUN |
| R-0 | | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 2-19. SYS Pin Control Register 1 (SYSPC1) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-1 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 0 | ECPCLKFUN | | ECLK function. This bit changes the function of the ECLK pin. |
| | | 0 | ECLK is in GIO mode. |
| | | 1 | ECLK is in functional mode as a clock output. |
| | | | **Note: Proper ECLK duty cycle is not guaranteed until 1 ECLK cycle has elapsed after switching into functional mode.** |

### 2.5.1.9 SYS Pin Control Register 9 (SYSPC9)

The SYSPC9 register, shown in Figure 2-16 and described in Table 2-27, controls the pull up/pull down configuration of the ECLK pin when it is configured as an input in GIO mode.

**Figure 2-16. SYS Pin Control Register 9 (SYSPC9) (offset = 20h)**

| 31 | | | 16 |
|---|---|---|---|
| Reserved | | | |
| R-0 | | | |
| 15 | | 1 | 0 |
| Reserved | | | ECPCLKPS |
| R-0 | | | R/W-0 |

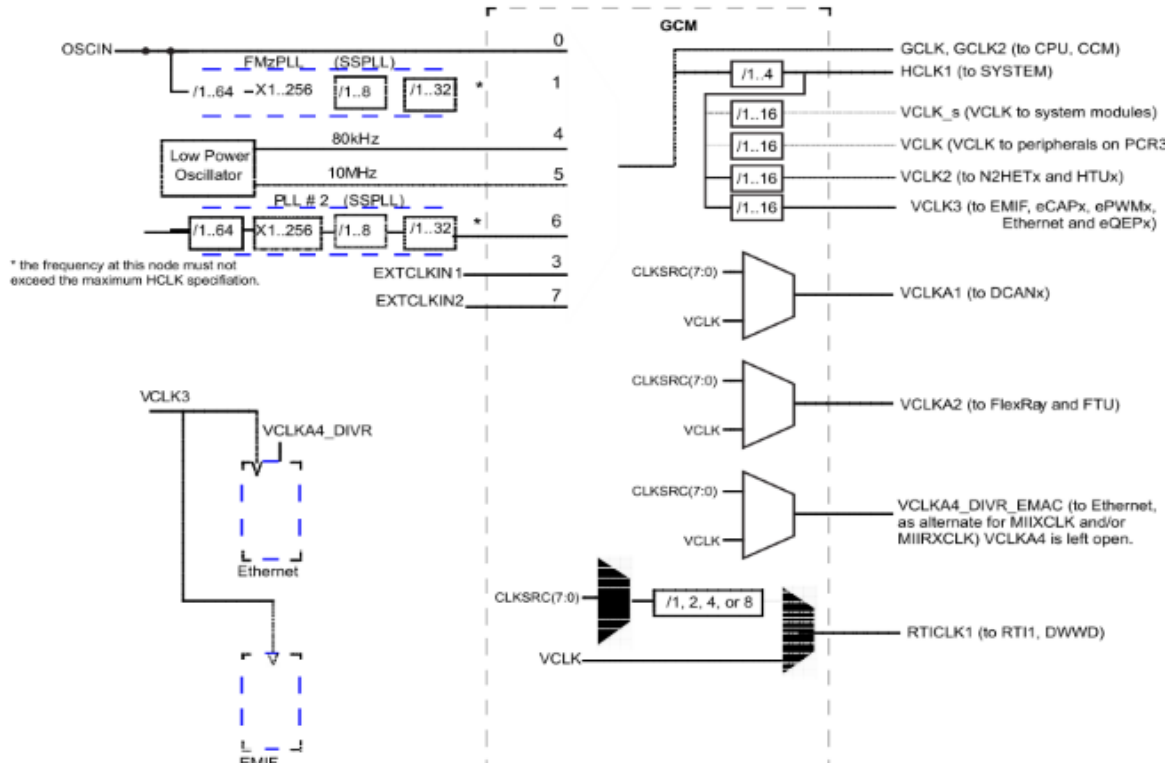
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 2-27. SYS Pin Control Register 9 (SYSPC9) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-1 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 0 | ECPCLKPS | | ECLK pull up/pull down select. This bit is only active when ECLK is configured as an input in GIO mode and the pull up/pull down logic is enabled. |
| | | 0 | ECLK pull down is selected, when pull up/pull down logic is enabled. |
| | | 1 | ECLK pull up is selected, when pull up/pull down logic is enabled. |
| | | | **Note: The ECLK pin pull up/pull down logic is enabled by clearing the ECPCLKPUE bit to 0 in the SYSPC8 register.** |
| | | | **Note: The ECLK pin is placed into GIO mode by clearing the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in input mode by clearing the ECPCLKDIR bit to 0 in the SYSPC2 register.** |

### 6.6.2.2 Mapping of Clock Domains to Device Modules

Each clock domain has a dedicated functionality as shown in Figure 6-6.




<CLOCK DOMAIN DDEVICE 회로>

systemREG1->ECPCNTL = (uint32)((uint32)0U << 24U)

| (uint32)((uint32)0U << 23U)

| (uint32)((uint32)(8U - 1U) & 0xFFFFU);

**Table 2-62. ECP Control Register (ECPCNTL) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-25 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 24 | ECPSSEL | | This bit allows the selection between VCLK and OSCIN as the clock source for ECLK.<br>**Note: Other ECLK clock sources are available for debug purposes by configuring the CLKTEST register.** |
| | | 0 | VCLK is selected as the ECP clock source. |
| | | 1 | OSCIN is selected as the ECP clock source. |
| 23 | ECPCOS | | ECP continue on suspend.<br>**Note: Suspend mode is entered while performing certain JTAG debugging operations.** |
| | | 0 | ECLK output is disabled in suspend mode. ECLK output will be shut off and will not be seen on the I/O pin of the device. |
| | | 1 | ECLK output is not disabled in suspend mode. ECLK output will not be shut off and will be seen on the I/O pin of the device. |
| 22-18 | Reserved | 0 | Reads return 0. Writes have no effect. |
| 17-16 | Reserved | 0 | Reads return 0 or 1 depends on what is written. The functionality of this bit is unavailable in this device. |
| 15-0 | ECPDIV | 0-FFFFh | ECP divider value. The value of ECPDIV bits determine the external clock (ECP clock) frequency as a ratio of VBUS clock or OSCIN as shown in the formula:<br>$ECLK = \frac{VCLK\ or\ OSCIN}{(ECPDIV + 1)}$ |

_coreEnableIrqVicOffset_

```
        mrc     p15, #0, r0,            c1, c0,   #0   //STCLR 을불러온다.
        orr     r0,   r0,      #0x01000000      //STCLR의 24번비트를 1로 설정
        mcr     p15, #0, r0,            c1, c0,   #0
        bx      lr

    .endasmfunc
```

FIQ로 쓸건지 IRQ로 쓸건지 결정하는 작업이 FIRQPR코드 이다.

급한 INTRRUPT는 FIQ로 해야 반응 빠르다.

```
/* set IRQ/FIQ priorities */
vimREG->FIRQPR0 = (uint32)((uint32)SYS_FIQ << 0U)
                | (uint32)((uint32)SYS_FIQ << 1U)
                | (uint32)((uint32)SYS_IRQ << 2U)
                | (uint32)((uint32)SYS_IRQ << 3U)
                | (uint32)((uint32)SYS_IRQ << 4U)
                | (uint32)((uint32)SYS_IRQ << 5U)
                | (uint32)((uint32)SYS_IRQ << 6U)
                | (uint32)((uint32)SYS_IRQ << 7U)
                | (uint32)((uint32)SYS_IRQ << 8U)
                | (uint32)((uint32)SYS_IRQ << 9U)
                | (uint32)((uint32)SYS_IRQ << 10U)
                | (uint32)((uint32)SYS_IRQ << 11U)
                | (uint32)((uint32)SYS_IRQ << 12U)
                | (uint32)((uint32)SYS_IRQ << 13U)
                | (uint32)((uint32)SYS_IRQ << 14U)
                | (uint32)((uint32)SYS_IRQ << 15U)
                | (uint32)((uint32)SYS_IRQ << 16U)
                | (uint32)((uint32)SYS_IRQ << 17U)
                | (uint32)((uint32)SYS_IRQ << 18U)
                | (uint32)((uint32)SYS_IRQ << 19U)
                | (uint32)((uint32)SYS_IRQ << 20U)
                | (uint32)((uint32)SYS_IRQ << 21U)
                | (uint32)((uint32)SYS_IRQ << 22U)
                | (uint32)((uint32)SYS_IRQ << 23U)
                | (uint32)((uint32)SYS_IRQ << 24U)
                | (uint32)((uint32)SYS_IRQ << 25U)
                | (uint32)((uint32)SYS_IRQ << 26U)
                | (uint32)((uint32)SYS_IRQ << 27U)
                | (uint32)((uint32)SYS_IRQ << 28U)
                | (uint32)((uint32)SYS_IRQ << 29U)
                | (uint32)((uint32)SYS_IRQ << 30U)
                | (uint32)((uint32)SYS_IRQ << 31U);
```

**_mpuInit_(); //mcu의 1:1매핑에 대해서 해킹하는것을 방지하는 부분이다. 모든 mcu는 mpu가 있다.**

**_mpuInit_**

```
; Disable mpu
mrc     p15, #0, r0, c1, c0, #0
bic     r0,   r0, #1 //2의배수로 정렬을 시킨다. 이말은 0번비트를 무조건 날린다.
dsb
mcr     p15, #0, r0, c1, c0, #0
isb //앞에 있는 명령어가 끝나기 전까지 아무것도 실행하지 말아라.
```

**<DSB>**

**이 부분은 사진을 보고**

**결론:메모리를 동기화하기전까지는 아무것도 하지 말라는 부분.**

```
 mrc     p15, #0, r0,          c1, c0, #0
         bic     r0,    r0, #0x20000     //17번비트를 날리겠다는 소리
         mcr     p15, #0, r0,          c1, c0, #0
```

```
mov     r0,    #0x0008
         orr     r0,    r0,       #0x1000
         mcr     p15, #0,       r0, c6, c1, #4
         movw    r0,    #((1 << 15) + (1 << 14) + (1 << 13) + (1 << 12) + (1 << 11) + (1 << 10) + (1 <<   9) + (1 <<   8) + (0x1F << 1) + (1))
         mcr     p15, #0,       r0, c6, c1, #2
```

**: 이 부분에서는 7번비트만 제외하고 모든 비트를 활성화 시키고 있는것이다. 7번비트를 아래그림에서 보면 읽을수도 쓸수도없는 상태이다. 결국 아에 건들지말라는 소리이다**

**_cacheEnable_**

| | | |
|---|---|---|
| [31:13] | - | SBZ. |
| [12] | XN | Execute Never. Determines if a region of memory is executable:<br>0 = all instruction fetches enabled<br>1 = no instruction fetches enabled. |
| [11] | - | Reserved. |
| [10:8] | AP | Access permission. Defines the data access permissions. For more information on AP bit values, see Table 4-38 on page 4-58. |
| [7:6] | - | SBZ. |
| [5:3] | TEX | Type extension. Defines the type extension attribute[a]. |
| [2] | S | Share. Determines if the memory region is Shared or Non-shared:<br>0 = Non-shared.<br>1 = Shared.<br>This bit only applies to Normal, not Device or Strongly Ordered memory. |
| [1] | C | C bit[a]: |
| [0] | B | B bit[a]: |

**_cacheEnable_**

```
stmfd sp!, {r0-r1}
mov r0,#0

MRC p15, #0, r1, c1, c0, #1        ; Read auxiliary control register
BIC r1, r1, #0x1 << 5              ; bit is default set to disable ECC. Clearing bit 5
MCR p15, #0, r1, c1, c0, #1        ; enable ECC, generate abort on ECC errors, enable
                                   ; hardware recovery

MRC    p15, #0, R1, c1, c0, #0     ; Read System Control Register configuration data
ORR    R1, R1, #0x1 <<12           ; instruction cache enable
ORR    R1, R1, #0x1 <<2            ; data cache enable
DSB
MCR    p15, #0, r0, c15, c5, #0    ; Invalidate entire data cache
DSB                                ; delay is required, manually added
```

```
MCR     p15, #0, r0, c7, c5, #0     ; Invalidate entire instruction cache
DSB                                 ; delay is required, manually added
MCR     p15, #0, R1, c1, c0, #0     ; enabled cache RAMs
ISB


ldmfd sp!, {r0-r1}


bx      lr
.endasmfunc.
```



### 19.9.11 Interrupt Enable Set Registers (REQENASET[0:3])

The interrupt register enable selectively enables individual request channels. Figure 19-27, Figure 19-28, Figure 19-29, Figure 19-30 and Table 19-16 describe these registers.

**NOTE:** Channel 0 and 1 are always enabled, not impacted by this register.

**Figure 19-27. Interrupt Enable Set Register 0 (REQENASET0) [offset = 30h]**

| 31 | 16 |
|---|---|
| REQENASET0[31:16] | |
| R/WP-0 | |

| 15 | 2 | 1 | 0 |
|---|---|---|---|
| REQENASET0[15:2] | | Reserved | |
| R/WP-0 | | R-3h | |

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

**Figure 19-28. Interrupt Enable Set Register 1 (REQENASET1) [offset = 34h]**

| 31 | 0 |
|---|---|
| REQENASET1[63:32] | |
| R/WP-0 | |

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

**Figure 19-29. Interrupt Enable Set Register 2 (REQENASET2) [offset = 38h]**

| 31 | 0 |
|---|---|
| REQENASET2[95:64] | |
| R/WP-0 | |

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

**Figure 19-30. Interrupt Enable Set Register 3 (REQENASET3) [offset = 3Ch]**

| 31 | 0 |
|---|---|
| REQENASET3[127:96] | |
| R/WP-0 | |

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

이 부분이 ISR[128]에 들어가는 값들이다.

```
vimREG->REQMASKSET0 = (uint32)((uint32)1U << 0U)
                    | (uint32)((uint32)1U << 1U)
                    | (uint32)((uint32)0U << 2U)
                    | (uint32)((uint32)0U << 3U)
                    | (uint32)((uint32)0U << 4U)
                    | (uint32)((uint32)0U << 5U)
                    | (uint32)((uint32)0U << 6U)
                    | (uint32)((uint32)0U << 7U)
                    | (uint32)((uint32)0U << 8U)
                    | (uint32)((uint32)0U << 9U)
                    | (uint32)((uint32)0U << 10U)
                    | (uint32)((uint32)0U << 11U)
                    | (uint32)((uint32)0U << 12U)
                    | (uint32)((uint32)0U << 13U)
                    | (uint32)((uint32)0U << 14U)
                    | (uint32)((uint32)0U << 15U)
                    | (uint32)((uint32)0U << 16U)
                    | (uint32)((uint32)0U << 17U)
                    | (uint32)((uint32)0U << 18U)
                    | (uint32)((uint32)0U << 19U)
                    | (uint32)((uint32)0U << 20U)
                    | (uint32)((uint32)0U << 21U)
                    | (uint32)((uint32)0U << 22U)
                    | (uint32)((uint32)0U << 23U)
                    | (uint32)((uint32)0U << 24U)
                    | (uint32)((uint32)0U << 25U)
                    | (uint32)((uint32)0U << 26U)
                    | (uint32)((uint32)0U << 27U)
                    | (uint32)((uint32)0U << 28U)
                    | (uint32)((uint32)0U << 29U)
                    | (uint32)((uint32)0U << 30U)
                    | (uint32)((uint32)0U << 31U);
```